Aggregate matrix-analytic techniques and their applications

Alma Riska
College of William & Mary - Arts & Sciences

Aggregate matrix-analytic techniques and their applications

A Dissertation

Presented to

The Faculty of the Department of Computer Science

The College of William & Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

Alma Riska

2002

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

[illegible]

A. Riska

Approved, November 2002

[illegible]

Evgenia Smirni
Thesis Advisor

[illegible]

Gianfranco Ciardo

Weizhen Mao

Weizhen Mao

[illegible]

Xiaodong Zhang

[illegible]

Mark Squillante
IBM T.J. Watson Research Center

To my father and the memory of my mother

Table of Contents

ACKNOWLEDGMENTS

This thesis would not have come to the form and shape it is today without the help and support of several people whom I would like to thank and to whom I am happily in debt.

First of all, I would like to thank my advisor, Dr. Evgenia Smirni for supporting, guiding, and working very closely with me for the last four years. This work started by researching one of her ideas and grew by continuous discussions with her. I am very grateful for her patience, her criticism, her persistence, and her friendly relationship with me.

I would like to thank Dr. Gianfranco Ciardo with whom I collaborated in several projects and whose knowledge and experience have helped me understand many problems better. I am grateful to Dr. Xiaodong Zhang and Dr. Weizhen Mao for being among my best teachers, for serving in my dissertation committee, and offering their support and assistance anytime I asked for it. Special thanks go to Dr. Mark Squillante, with whom I worked closely during the summer of 2001, for helping me approach a broader and more challenging set of problems and solution techniques. I would like to thank Wei Sun, Vesselin Diev, Qi Zhang, and Daniella Puiu for helping me obtain analysis results presented in various parts of this dissertation.

I am very grateful to the entire faculty and staff of the Computer Science Department for their encouragement every step of the way, for always being friendly, helpful, and appreciative of our work. In particular, I would like to thank Vanessa Godwin for supporting us in so many ways and making it easier for us to concentrate in our work and studies. During my graduate studies, I was supported by the Computer Science Department and by the National Science Foundation, through my advisor's grants, and I would like to acknowledge them for their financial support.

My thesis dedication and my most wholehearted thanks go to my parents, Pandi and Marjeta Riska. Without their unconditional love, support, and guidance, I would have never come this far. I want to thank my sisters, Zhuljeta and Mimoza, and my brother, Andrea, for supporting me on all decisions that I have made in my life.

My most special thanks go to my friend and long-time partner, Artan Dimnaku, for supporting me throughout these four not-that-easy years. I am lucky to have had him by my side to advise me, understand me, and unconditionally love me.

List of Tables

List of Figures

ABSTRACT

The complexity of computer systems affects the complexity of modeling techniques that can be used for their performance analysis. In this dissertation, we develop a set of techniques that are based on tractable analytic models and enable efficient performance analysis of computer systems. Our approach is three pronged: first, we propose new techniques to parameterize measurement data with Markovian-based stochastic processes that can be further used as input into queueing systems; second, we propose new methods to efficiently solve complex queueing models; and third, we use the proposed methods to evaluate the performance of clustered Web servers and propose new load balancing policies based on this analysis.

We devise two new techniques for fitting measurement data that exhibit high variability into Phase-type (PH) distributions. These techniques apply known fitting algorithms in a divide-and-conquer fashion. We evaluate the accuracy of our methods from both the statistics and the queueing systems perspective. In addition, we propose a new methodology for fitting measurement data that exhibit long-range dependence into Markovian Arrival Processes (MAPs).

We propose a new methodology, ETAQA, for the exact solution of M/G/1-type processes, GI/M/1-type processes, and their intersection, i.e., quasi birth-death (QBD) processes. ETAQA computes an aggregate steady state probability distribution and a set of measures of interest. ETAQA is numerically stable and computationally superior to alternative solution methods. Apart from ETAQA, we propose a new methodology for the exact solution of a class of GI/G/1-type processes based on aggregation/decomposition.

Finally, we demonstrate the applicability of the proposed techniques by evaluating load balancing policies in clustered Web servers. We address the high variability in the service process of Web servers by dedicating the servers of a cluster to requests of similar sizes and propose new, content-aware load balancing policies. Detailed analysis shows that the proposed policies achieve high user-perceived performance and, by continuously adapting their scheduling parameters to the current workload characteristics, provide good performance under conditions of transient overload.

Aggregate matrix-analytic techniques and their applications

Chapter 1

Introduction

As Internet continues to grow, scientists and practitioners are faced with the challenge to design and manage systems that keep up with the ever increasing number of users and complexity of services. In such an environment, analyzing and predicting the performance of existing and future systems is of great importance. Nevertheless, the task of modeling the performance of Internet-related systems is non-trivial, as their inherent complexity severely limits the set of available tools for their analysis.

There is ample evidence [4, 3, 7, 52] that the arrival and service processes in Internet-related systems exhibit long-range dependence and high variability, respectively. These characteristics impact the performance of such systems [30, 14] considerably and, unfortunately, increase the complexity of simulation and modeling tools that can be used for their analysis. Specifically, simulations need to run for long time to capture the effects of "rare" events associated with high variability, while the traditional analytic modeling techniques result in models that are intractable.

In this dissertation, we focus on the development of new analytic techniques that allow for the *efficient* and *accurate* performance analysis of Internet-related systems. We focus on models for Internet-related systems and workloads, and their interaction. The objective, graphically depicted in Figure 1.1, is to propose techniques and tools that can be used in the process of improving the performance of Internet-related systems. We achieve this objective by proposing Markovian-based models for the workload and complex queueing models for the systems. The systems performance is analyzed by solving efficiently the proposed queueing models. Finally, based on the results of our analysis, we propose new resource allocation policies that aim at improving the user perceived performance.

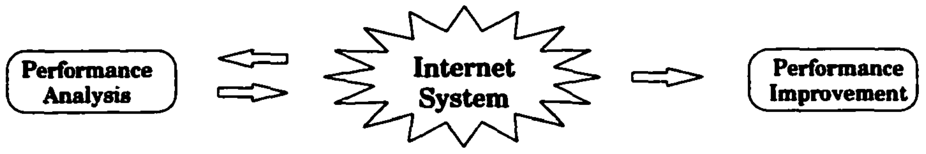

Figure 1.1: Our objective: improve system performance via performance analysis.

Our techniques are based on the **matrix-analytic** methodology. Matrix-analytic methods provide flexible tractable models for the analysis of systems with complex behavior. The matrix analytic methodology, pioneered by Marcel Neuts [67, 69], is advanced by a considerable body of research [46, 75, 76, 47, 60], and is applied in many fields of performance evaluation of computer and communication systems [66, 65, 44, 33, 49, 102, 103, 100, 106, 1] as well as in other scientific areas [63].

1.1 Contributions

The focus of this dissertation is three-fold:

- to characterize and approximate the arrival and service processes in Internet-related systems by analytic models that allow the use of matrix-analytic methods for their performance analysis,
- to propose *new* aggregation-based matrix analytic methodologies for the efficient solution of GI/M/1-type, M/G/1-type, and GI/G/1-type Markov processes, and
- to present a case-study where we directly apply these methodologies.

The service process in Internet-related systems is best characterized by heavy-tailed distributions [3, 7, 8, 95]. These distributions are intractable since usually their moment generating functions or Laplace transforms cannot be derived in closed form formulas. Moreover, the performance of queueing systems whose service processes are best described by such distributions is affected considerably, since measures like average queue length, average response time, and their respective distributions inherit the heavy-tailed behavior [30].

In this dissertation, we propose parameterization techniques to capture the high-variable behavior of the service process and the long-range behavior of the arrival process in Internet-related systems using tractable distributions such as Phase-type (PH) distributions and Markovian Arrival Processes (MAPs), respectively. Although the Markov property suggests lack of memory and independence in the stochastic process, MAPs and PH distributions accurately capture properties such as long-range dependence and high variability, respectively. Our parameterization techniques apply on measurement data rather than distribution functions and one of their important features is that they scale well with the size of data sets.

The major contribution of the dissertation is a new analytic methodology called ETAQA, which is an aggregation-based methodology that solves efficiently queueing systems whose embedded Markov chains have a repetitive structure, i.e., M/G/1-type and GI/M/1-type patterns. ETAQA allows for the exact computation of an *aggregate* steady-state probability vector of such processes, as well as for the exact computation of a wide set of measures of interest. The aggregation process in ETAQA does not follow the rules of classic aggregation as described by Courtois [27]. Instead, we exploit the structure of the embedded Markov chain and aggregate using global balance equations. ETAQA provides exact solutions that are numerically stable and computationally efficient, when compared with alternative solution methods.

Apart from ETAQA, we also present an aggregation-based technique that solves a broader class of stochastic processes whose embedded Markov chains have a repetitive structure of the GI/G/1-type. In this method, which applies only for a restrictive class of GI/G/1-type processes, we exploit the special structure of the embedded Markov chain and define an efficient and exact solution for the computation of the steady-state probability distribution of the process.

We devise new load balancing policies for clustered Web servers and evaluate their performance using the proposed analytic methodologies. We use our fitting techniques as well as existing ones [6, 30] to characterize and parameterize the service process in clustered Web servers. The models obtained from the fittings are used as inputs to queueing models. We use ETAQA to solve these queueing models. The set of performance metrics obtained from the solution of Web server queueing models, allows us to identify problems and to offer solutions to further improve cluster performance. We focus on load balancing policies that use the workload characteristics, e.g., variability and arrival intensity, to compute their balancing parameters. We propose two new load balancing policies, EQUILOAD and ADAPTLOAD, for clustered Web servers. These policies manage well cluster resources as they quickly respond to changes in the cluster operating conditions.

1.2 Organization

This document is organized as follows. The basic concepts on stochastic and queueing models, Markov chains, and aggregation solution techniques are reviewed in Chapter 2. A summary of prior work on matrix analytic methods is given in Chapter 3. The new fitting techniques that approximate long-tailed and long-range dependent data sets with PH distributions and Markovian Arrival Processes, respectively, are presented in Chapter 4. In Chapter 5, the new aggregate matrix-analytic methodology, ETAQA, is presented and its performance is compared with the classic matrix-analytic techniques. Chapter 6 presents a methodology for the exact solution of a class of GI/G/1-type processes. In Chapter 7, we focus on the applicability of our methodology in load balancing of clustered Web servers. This chapter gives a closer look in the characterization of the workload and demonstrates how analysis can be more efficient when the proposed methodology is used. Finally, Chapter 8 outlines our contributions and gives directions for future work.

Chapter 2

Background

This chapter presents an overview of the basic concepts, terminology, and related work regarding the problems that we address in this dissertation. Since our focus is on matrix analytic methods, whose fundamentals are on Markov chains, we present basic definitions, concepts, and solution methods for Markov chains. We identify the infinite Markov chains with repetitive structures that we focus on. We also describe PH-type distributions and Markovian Arrival Processes as the stochastic processes most commonly associated with matrix-analytic methods.

This chapter is organized as follows. In Section 2.1, we introduce basic notation. In Section 2.2, we define stochastic processes in general and focus on the exponential distribution and its variations that we use throughout this dissertation. We give an overview of the definition and classifications of Markov chains in Section 2.4. In Section 2.5, we present examples and define the infinite Markov chains with repetitive structures. We describe the related work on aggregation/decomposition techniques for solution of Markov chains in Section 2.8.

2.1 Basic notation

We use:

- calligraphic letters to indicate sets (e.g., $\mathcal{A}$),
- lower case boldface Roman or Greek letters to indicate vectors (e.g., $\mathbf{a}$, $\boldsymbol{\alpha}$),
- upper case boldface Roman letters to indicate matrices (e.g., $\mathbf{A}$),
- superscripts in parentheses or subscripts to indicate family of related entities (e.g., $\mathcal{A}^{(1)}$, $\mathbf{A}_{\mathcal{A}_1}$),
- square brackets to indicate vector and matrix elements (e.g., $\mathbf{a}[1]$, $\mathbf{A}[1,2]$),
- sets of indices within square brackets to indicate subvectors or submatrices (e.g., $\mathbf{a}[\mathcal{A}]$, $\mathbf{A}[\mathcal{A},\mathcal{B}]$),
- $RowSum(\cdot)$ to indicate the diagonal matrix whose entry in position (r,r) is the sum of the entries on the r^{th} row of the argument (which can be a rectangular matrix),
- $Norm(\cdot)$ to indicate a matrix whose rows are normalized,
- $\mathbf{0}$ to indicate a row vector or a matrix of 0's of the appropriate dimensions,
- $\mathbf{1}$ to indicate a row vector of 1's, of the appropriate dimension,

- $\boldsymbol{\pi}$ to indicate the steady state or the stationary probability distribution of a stochastic process[1],

- $\mathbf{P}$ to indicate the probability transition matrix of a discrete time Markov chain (DTMC)[2] unless differently specified, and

- $\mathbf{Q}$ to indicate the infinitesimal generator matrix of a continuous time Markov chain (CTMC)[3].

2.1.1 Kendall Notation

We use Kendall notation to describe the characteristics of the queueing systems that we focus on. The notation is of the form:

$$A/S/s/c/p/D$$

where

- A stands for the description of the arrival process, (e.g., M stands for Markovian interarrivals, GI for general (any distribution) independent arrivals, MAP for arrivals driven by a Markovian Arrival Process).

- S stands for the service time distribution, (e.g., M stands for Markovian service, G for general (any distribution) service, PH for phase-type service).

[1]Definition of stationary distribution vector is given in Section 2.4.
[2]Definition of a DTMC and $\mathbf{P}$ is given in Section 2.4.
[3]Definition of a CTMC and $\mathbf{Q}$ is given in Section 2.4.

- s stands for the number of servers in the system and can be any integer equal or larger than 1.

- c stands for the capacity of the queue, i.e., the maximum number of jobs that can be queued in the system ($c \geq 0$). If this argument is missing, then, by default, the queue capacity is infinity.

- p stands for the system population, i.e., the maximum number of jobs that can arrive in the queue. If this argument is missing then, by default, the system population is infinity.

- D stands for the queueing discipline, which can be FIFO (first come first serve), LIFO (last come first serve), or any other queueing discipline. If this argument is missing, then, by default, the queueing discipline is FIFO.

The simplest and the easiest queueing system to analyze is the M/M/1 queue, where job interarrivals and service times are Markovian and there is a single server in the system (missing arguments in this notation of the queuing system means that they take the default values). If the single server admits Markovian arrivals, but the service process is governed by a general distribution, then the description of the queue using Kendall notation is M/G/1.

2.2 Random variables

In practice, if the outcome of a process is *not* known in advance, then the process is nondeterministic or *stochastic*. A *stochastic process* is the set of outcomes of a random occurrence of a process, indexed by time [65]. *State* is the condition of a stochastic process, in a specific time, described by means of *random variables*. The *state space S* is the set of all possible states of a stochastic process. Most of the stochastic models are expressed in terms of *random variables*: the number of jobs in the queue, the faction of time a processor is busy, the amount of time a server is operational are examples of random variables. Random variables are functions that map a real value to every random outcome of the state space. Random variables that take a countable number of values are *discrete*, otherwise they are *continuous* [65].

Given that X is a random variable, there are two important functions that characterize X, i.e., the cumulative distribution function (CDF) and the probability density function (PDF). The cumulative distribution function $F(x)$ is nondecreasing such that $F(-\infty) = 0$ and $F(\infty) = 1$. If X is a continuous random variable, the probability density function $f(x)$ is nonnegative and $\int_{-\infty}^{\infty} f(z)dz = 1$. If X is a discrete random variable, then the PDF $f(x)$ is nonnegative and $\sum_{z=-\infty}^{\infty} f(z) = 1$. If the random variable X is discrete (continuous) then its PDF $f(x)$ and its CDF $F(x)$ are also discrete (continuous).

The *expectation* of a continuous random variable X is defined by $E[X] = \int_{-\infty}^{\infty} zf(z)dz$, provided the integral converges absolutely. If the random variable X is discrete then its expectation is defined by $E[X] = \sum_{z=-\infty}^{\infty} zf(z)$. More generally one can define the expectation of a function $h(X)$ of a continuous random variable X by $E[h(X)] = \int_{-\infty}^{\infty} h(z)f(z)dz$. The expectation of a function of the form $h(x) = x^k$, for $k \geq 1$ is known as the k^{th} *moment* of a distribution. Similar definitions exist for discrete random variables.

Other metrics that are often used in performance analysis are the standard deviation, coefficient of variation, and median. The *standard deviation* (STD) and *coefficient of variation* (CV) are two metrics that quantify the variability of a random variable X, and are defined as $STD = \sqrt{E[X^2] - E[X]^2}$ and $CV = STD/E[X]$, respectively [48]. Median of a random variable, which measures central tendency in a random variable X, is the smallest value x such that the CDF $F(x) \geq 0.5$ [48].

2.2.1 Exponential distribution

One of the most studied continuous distributions is the *exponential* distribution. It is widely used to describe various stochastic processes. An important property of the exponential distribution is the *memoryless* property. This property plays an important role in computer system modeling since the value of an exponential random variable in a given moment of time t does not depend on its past values, therefore it allows us to analyze the system without keeping track of all past events. The cumulative distribution function and the probability density distribution function of an exponential random variable X are:

$$F_{E_\lambda}(x) = \begin{cases} 0 & x < 0, \\ 1 - e^{-\lambda x} & 0 \le x, \end{cases} \qquad f_{E_\lambda}(x) = \begin{cases} 0 & x < 0, \\ \lambda \cdot e^{-\lambda x} & 0 \le x, \end{cases} \tag{2.1}$$

where λ is the parameter that characterizes the distribution, and E_λ denotes an exponential distribution with parameter λ.

2.2.2 Variations of exponential distribution

Let E_{λ_i} $i = 1, ..., n$ be n independent exponential random variables each with parameter λ_i $i = 1, ..., n$, where $\lambda_i \neq \lambda_j$ for $i \neq j$. Suppose that there are n positive constants p_i, $i = 1, ..., n$ such that $\sum_{i=1}^{n} p_i = 1$. If the random variable $X = E_{\lambda_i}$ with probability p_i, then X is a *hyperexponential* random variable with n exponential stages and parameters p_i, λ_i, $i = 1, ..., n$ [88]. More formally, let the hyperexponential random variable X be denoted by $Hr(n)$ and define its PDF as.

$$f_X = f_{Hr(n)} = \sum_{i=1}^{n} p_i \cdot f_{E_{\lambda_i}}. \tag{2.2}$$

The hyperexponential distribution $Hr(n)$ has $2n-1$ parameters, while the exponential distribution E_λ has only one.

Let E_{λ_i} $i = 1, ..., n$ be n independent exponential random variables each with parameter λ_i for $i = 1, ..., n$, such that $\lambda_i \neq \lambda_j$ for $i \neq j$. The random variable X has a *hypoexponential* distribution if

$$X = Hypo(n) = \sum_{i=1}^{n} X_i. \tag{2.3}$$

The probability density function of a $Hypo(n)$ is not trivial to compute (see [88] for more details). The hypoexponential distribution has only n parameters. A special case of a hypoexponential distribution arises when $\lambda_i = \lambda, \forall i = 1, \dots, n$ and is known as the *Erlang* distribution. The Erlang distribution is denoted by $Er(n)$ and characterized by λ and n.

It is easy to observe that the hypoexponential and hyperexponential distributions consist of several exponential "stages" (or "phases"). If, instead of exponential stages in series (hypoexponential) or stages in parallel (hyperexponential), arbitrary interaction among the collection of exponential stages is allowed, then the result is the PH distribution. We will discuss PH distributions in more detail in Section 2.6.

2.3 Heavy tailed distributions

A class of distributions that is often used to capture the characteristics of highly-variable stochastic processes, i.e., more variable than the exponential distribution, is the class of heavy-tailed distributions.

Definition A distribution is heavy-tailed if its *complementary cumulative distribution* (CCDF), often referred to as the tail, $F^c(t) = 1 - F(t)$, where

$F(t)$ is the CDF, decays slower than exponentially, i.e., there is some $\gamma > 0$ such that

$$\lim_{t \to \infty} \exp(\gamma t) F^c(t) \to \infty. \tag{2.4}$$

A typical heavy-tailed distribution is *power-tailed* if $\mathbf{F}^c(t) \sim \alpha t^{-\beta}$ as $t \to \infty$ for constants $\alpha > 0$ and $\beta > 0$.

Definition A distribution has short tail if its CCDF $F^c(t)$, decays exponentially or faster, i.e., there is some $\gamma > 0$ such that

$$\lim_{t \to \infty} \exp(\gamma t) F^c(t) \to 0. \tag{2.5}$$

A typical short-tailed distribution is *exponentially-tailed* if $\mathbf{F}^c(t) \sim \alpha e^{-\beta t}$ as $t \to \infty$ for constants $\alpha > 0$ and $\beta > 0$.

In the literature, different definitions of heavy-tailed like distributions exist. For a more detailed classification of heavy-tailed distributions and their properties refer to [95] and references therein. Throughout this dissertation, we refer to a distribution as heavy-tailed if its coefficient of variation is larger than the one of the exponential distribution. Note that we distinguish between the unbounded possible values of a distribution function and the bounded possible values in a data set by referring to high variability in a data set as long-tailed behavior and in a distribution function as heavy-tailed behavior.

The *Pareto* distribution with $F^c_{Pareto(\beta)}(t) = t^{-\beta}$ for $\beta > 0$ is a classic case of a distribution exhibiting power-tailed behavior in the entire range of its parameters. The *Weibull* distribution with $F^c_{Weibull(c,a)}(t) = e^{-(t/a)^c}$ for $c < 1$ and $a > 0$ is heavy-tailed, but not power-tailed.

Simulation of heavy-tailed distributions for estimation of steady-state measures is not easy, as the simulation must run exceptionally long in order to capture the effect of the distribution tail, i.e., the rare events, which even with small probability of occurrence can significantly affect the system performance. Heavy-tailed distributions generally have high coefficient of variation, while power-tailed distributions can have infinite moments. For example, in a power-tailed distribution if $0 < \beta < 1$ the mean is infinite, while if $1 < \beta < 2$ the mean of the distribution is finite but the variance is infinite. The infinite mean or variance in a power-tailed distribution increases the complexity of their analysis. The highly variable behavior in data sets or distribution functions can be accurately approximated by PH distributions (PH distributions are discussed later in this chapter), which are tractable and can be analyzed using the matrix-analytic methodology.

2.4 Markov processes

A *Markov process*[4] is a stochastic process that has a limited form of "historical" dependency [65]. Let $\{X(t) \ : \ t \in \mathcal{T}\}$ be defined on the parameter set $\mathcal{T}$ and assume that it represents time. The values that $X(t)$ can obtain are called *states*, and all together they define the *state space* $\mathcal{S}$ of the process. A stochastic process is a Markov process if it satisfies

$$P[X(t_0+t_1) \leq x \mid X(t_0) = x_0, \ X(\tau), -\infty < \tau < t_0] = P[X(t_0+t_1) \leq x \mid X(t_0) = x_0], \ \forall t_0, t_1 > 0. \tag{2.6}$$

Let t_0 be the present time. Eq.(2.6) states that the evolution of a Markov process at a future time, conditioned on its present and past values, depends only on its present value [65]. The condition of Eq.(2.6) is also known as the *Markov property*. Markov chains are classified as discrete or continuous.

2.4.1 Discrete time Markov chains (DTMCs)

Consider a Markov process as defined by Eq.(2.6) and, without loss of generality, let the state space $\mathcal{S}$ be the set of nonnegative integers. The DTMC that characterizes the process captures its evolution among states of $\mathcal{S}$ over time $t \in \mathcal{T}$. The *transition probability* between state i to state j at time $n-1$ is the probability $P[X_n = j \mid X_{n-1} =$

[4]Named after A.A. Markov (1856-1922), a Russian mathematician who made fundamental contributions to probability theory.

$i]$. A DTMC is *time homogeneous* if

$$P[X_n = j \mid X_{n-1} = i] = P[X_{m+n} = j \mid X_{m+n-1} = i], \quad n = 1, 2, \ldots, \quad m \geq 0, \quad i, j \in \mathcal{S}. \tag{2.7}$$

Throughout this dissertation, we consider time homogeneous Markov chains. Further we define $p_{i,j} \overset{\text{def}}{=} P[X_n = j \mid X_{n-1} = i]$ and the one step *probability transition* matrix $\mathbf{P}$:

$$\mathbf{P} \overset{\text{def}}{=} \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,j} & \cdots \\ p_{1,0} & p_{1,1} & \cdots & p_{1,j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ p_{i,0} & p_{i,1} & \cdots & p_{i,j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}. \tag{2.8}$$

Each row of the probability transition matrix represents the transition flow out of the corresponding state. Each column of it represents the transition flow into the state. As the cumulative transition flow out of each state must be 1, the rows of matrix $\mathbf{P}$ must sum to 1 and have all non-negative elements (since they are probabilities). A matrix that has non-negative elements and its rows sum to 1 is often called a *stochastic matrix*. If $\mathcal{S}$ is finite, then $\mathbf{P}$ has finite dimension.

The probability that the DTMC reaches, on the n^{th} step, state j starting from state i in step 0 is given by the *Chapman-Kolmogorov equation*:

$$p^n_{i,j} = \sum_{k \in \mathcal{S}} p^m_{i,k} \cdot p^{n-m}_{k,j} \quad 0 \leq m \leq n. \tag{2.9}$$

Let $\boldsymbol{\pi}^n \overset{\text{def}}{=} (\pi^n_0, \pi^n_1, \ldots)$ be the probability vector whose element π^n_i denotes the probability that the DTMC is at state i at step n. Since $\boldsymbol{\pi}^n = \boldsymbol{\pi}^0 \cdot \mathbf{P}^n$, $n = 1, 2, \ldots$, then the probability of state i at step n is simply the sum of the probabilities along all sample paths from j to i in n steps weighted by the probability of starting at state j [65].

A DTMC is *irreducible* if for each pair of states $(i, j) \in \mathcal{S}^2$ there exist an integer n such that $p^n_{i,j} > 0$ [43]. State i is *positive recurrent* if $\sum_{n \in \mathbf{N}} p^n_{i,i} = \infty$. A state is *periodic* if $p^n_{i,i} > 0$ iff $n = k \cdot d$ for some values of k and a fixed value of $d > 1$. If a state is not periodic. then it is *aperiodic*. A state is called *ergodic* if it is positive recurrent and aperiodic.

The following two propositions are fundamental for the analysis of DTMCs.

Proposition[65] If $\mathbf{P}$ is the probability transition matrix of an irreducible DTMC in an ergodic set of states. the limiting matrix $\lim_{n \to \infty} \mathbf{P}^n$ has identical rows that are equal to the *stationary probability vector* $\boldsymbol{\pi} = \lim_{n \to \infty} \boldsymbol{\pi}^n$.

Proposition[65] The stationary probability vector of an irreducible DTMC in an ergodic set of states is unique and satisfies

$$\boldsymbol{\pi} = \boldsymbol{\pi} \cdot \mathbf{P} \tag{2.10}$$

and the normalization condition

$$\boldsymbol{\pi} \cdot \mathbf{1}^T = 1. \tag{2.11}$$

We always refer to an ergodic and irreducible Markov chain throughout this dissertation, and are interested on computing the stationary distribution vector $\boldsymbol{\pi}$ which characterizes the *steady state* probability distribution of the process. The steady state is reached after the process passes an arbitrary large number of steps. In steady state, the total flow out of a state is equal to the total flow into the state. This property is called *flow balance* and is expressed in the form of a *flow balance equation*. The collection of all flow balance equations for a DTMC is formally represented by Eq.(2.10).

2.4.2 Continuous time Markov chains (CTMCs)

In a CTMC, the process makes a transition from one state to another, after it has spent *an amount of time* on the state it starts from. This amount of time is defined as the *state holding time*. In a DTMC the holding time is geometrically distributed, while in a CTMC it is exponentially distributed. All DTMC definitions apply for CTMCs as well. In the same way that we build the probability transition matrix for a DTMC, we construct the *infinitesimal generator matrix* $\mathbf{Q}$ of a CTMC. The entries of the infinitesimal generator matrix $\mathbf{Q}$ are the rates at which the process jumps from state to state. By definition, the diagonal entries of $\mathbf{Q}$ are equal to minus the total rate out of the state that corresponds to that row, $q_{i,i} = -\sum_{j \neq i}^{\infty} q_{i,j}$. This implies that the row sums of $\mathbf{Q}$ equal 0:

$$\mathbf{Q} \overset{\text{def}}{=} \begin{bmatrix} -q_0 & q_{0.1} & \cdots & q_{0.j} & \cdots \\ q_{1.0} & -q_1 & \cdots & q_{1.j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ q_{i.0} & q_{i.1} & \cdots & q_{i.j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}. \tag{2.12}$$

Similar to DTMCs, the following proposition holds for CTMCs.

Proposition[65] The stationary probability vector π of an irreducible CTMC in an ergodic set of states is unique and satisfies

$$\pi \cdot \mathbf{Q} = 0 \tag{2.13}$$

and the normalization condition

$$\pi \cdot \mathbf{1}^T = 1. \tag{2.14}$$

When the CTMC process is in steady state, the property of flow balance holds, and Eq.(2.13) represents all the flow balance equations of the CTMC.

2.5 Markov chains with repetitive structure

Markov chains with repetitive structure occur when queueing models are analyzed via their embedded Markov chains. The simplest queueing system is the M/M/1 queue. According to Kendal notation in a M/M/1 queue both arrival and service processes are Markovian. Since there is no restriction on the system population and queue capacity, the state space of the embedded CTMC is infinite and its state transition diagram is illustrated in Figure 2.1. Usually, an M/M/1 queue is known as a *birth-death* stochastic process.

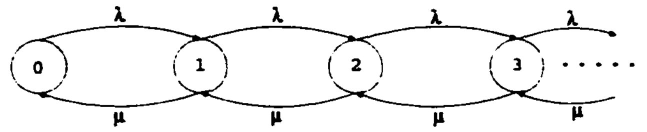

Figure 2.1: The state transition diagram of an M/M/1 queue.

In the CTMC of Figure 2.1, each state of the system is distinguished by the number of jobs in the system. The state space of the process is $\mathcal{S} = \{0, 1, 2, ...\}$. If the mean arrival rate of jobs in the system is λ and the mean service rate is μ, then the following geometric relation holds for the elements of the stationary probability vector $\boldsymbol{\pi}$: $\pi_i = \pi_{i-1}\frac{\lambda}{\mu}$, $\forall i \geq 1$, where π_i is the steady state probability of state i [43]. If we order the states of the chain in an increasing order, then the infinitesimal generator matrix of the M/M/1 queue is

$$\mathbf{Q}_{M/M/1} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ \mu & -(\lambda+\mu) & \lambda & 0 & 0 & \cdots \\ 0 & \mu & -(\lambda+\mu) & \lambda & 0 & \cdots \\ 0 & 0 & \mu & -(\lambda+\mu) & \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (2.15)$$

2.5.1 Quasi birth-death processes

Consider now a single queue that accepts Poisson arrivals at rate λ. Jobs require two exponential stages of service: the first at rate μ_1 and the second at rate μ_2. The service time distribution as described is a hypoexponential distribution with 2 phases, i.e.,

$Hypo(2)$. This is an M/Hypo(2)/1 queue. Each state is defined by (i, s), where i is the number of jobs in the queue and s is the stage of service for the job being served. If we order states lexicographically, then $\mathcal{S} = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,1), (2,2), ...\}$ and the infinitesimal generator matrix $\mathbf{Q}_{M/Hypo(2)/1}$ of the process is

$$\mathbf{Q}_{M/Hypo(2)/1} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & -\alpha_1 & \mu_1 & \lambda & 0 & 0 & 0 & \cdots \\ \mu_2 & 0 & -\alpha_2 & 0 & \lambda & 0 & 0 & \cdots \\ 0 & 0 & 0 & -\alpha_1 & \mu_1 & \lambda & 0 & \cdots \\ 0 & \mu_2 & 0 & 0 & -\alpha_2 & 0 & \lambda & \cdots \\ 0 & 0 & 0 & 0 & 0 & -\alpha_1 & \mu_1 & \cdots \\ 0 & 0 & 0 & \mu_2 & 0 & 0 & -\alpha_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad (2.16)$$

where $\alpha_j \overset{\text{def}}{=} \lambda + \mu_j$, $j = 1, 2$. The state transition diagram of this process is illustrated in Figure 2.2. Note that by grouping the entries of the generator matrix of Eq.(2.16) according to the number of jobs in the system, then the structure of the new block partitioned Eq.(2.16) closely resembles the structure of the generator matrix of an M/M/1 queue, shown in Eq.(2.15).

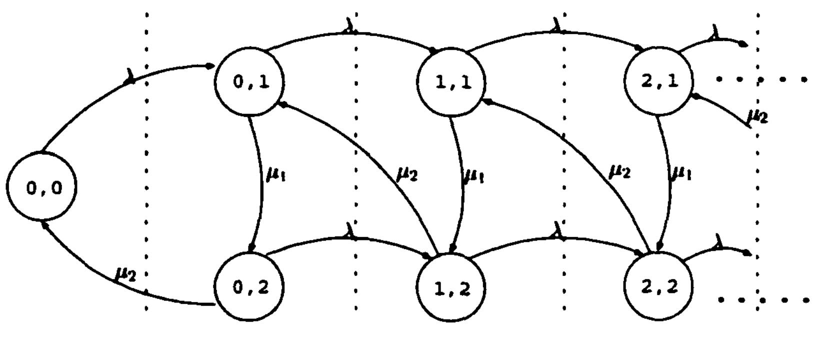

Figure 2.2: The state transition diagram of a M/Hypo2/1 queue.

We define subvectors of the stationary distribution vector $\boldsymbol{\pi}$ of the process as $\boldsymbol{\pi}^{(i)} \stackrel{\text{def}}{=} (\boldsymbol{\pi}_{(i,1)}, \boldsymbol{\pi}_{(i,2)})$ for $i \geq 1$, and $\boldsymbol{\pi}^{(0)} \stackrel{\text{def}}{=} (\boldsymbol{\pi}_{(0,0)}, (\boldsymbol{\pi}_{(0,1)}, \boldsymbol{\pi}_{(0,2)})$. Accordingly, we define subsets $\mathcal{S}^{(i)}$ for $i \geq 1$ and $\mathcal{S}^{(0)}$ of the state space $\mathcal{S}$. We call the states on $\mathcal{S}^{(0)}$ the *boundary portion* of the process and all other states are called the *repeating portion* or the *repeating states* of the process[5]. Throughout this dissertation, it is assumed that the cardinality of the boundary portion of the process is m and the cardinality of each repeating level is n, where m may be different from n. This partition of the stationary distribution vector $\boldsymbol{\pi}$ and state space $\mathcal{S}$ of the process defines the following submatrices of the infinitesimal generator in Eq.(2.16).

$$
\begin{gathered}
\mathbf{F} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} -\alpha_1 & \mu_1 \\ 0 & -\alpha_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ \mu_2 & 0 \end{bmatrix}, \\
\widehat{\mathbf{L}} = \begin{bmatrix} -\lambda' & \lambda' & 0 \\ 0 & -\alpha_1 & \mu_1 \\ \mu_2 & 0 & -\alpha_2 \end{bmatrix}, \quad \widehat{\mathbf{F}} = \begin{bmatrix} 0 & 0 \\ \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \quad \widehat{\mathbf{B}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mu_2 & 0 \end{bmatrix}.
\end{gathered}
\tag{2.17}
$$

Based on the above definitions, we can block partition the infinitesimal generator of Eq.(2.16) as follows

$$
\mathbf{Q}_{QDB} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{2.18}
$$

[5]In the literature, the boundary and the repeating portion of an infinite CTMC are also referred to as the non-homogeneous and the homogeneous parts of the CTMC, respectively [32].

where **0** is a matrix of all zeros of the appropriate dimension. The letters "L", "F", and "B" are used to denote the "local", "forward", and "backward" transition rates, respectively, in relation to a set of states $\mathcal{S}^{(j)}$ for $j \geq 1$. We use "$\widehat{\ }$" for matrices related to $\mathcal{S}^{(0)}$. The similarity of infinitesimal generator in Eq.(2.18) with the infinitesimal generator in Eq.(2.15) gives such processes the name "quasi birth-death" (QBD). We discuss the solution methods for QBD processes in Chapter 3.

2.5.2 GI/M/1-type processes

Consider a single server queue where failures can happen during service. The occurrence of failures flushes the queue. Suppose that the arrivals are Markovian with rate λ and the service process is two exponential stages with rates μ_1 and μ_2 respectively. Failures occur exponentially with rate f. The state transition diagram of this process is illustrated in Figure 2.3. The block partitioned infinitesimal generator for the

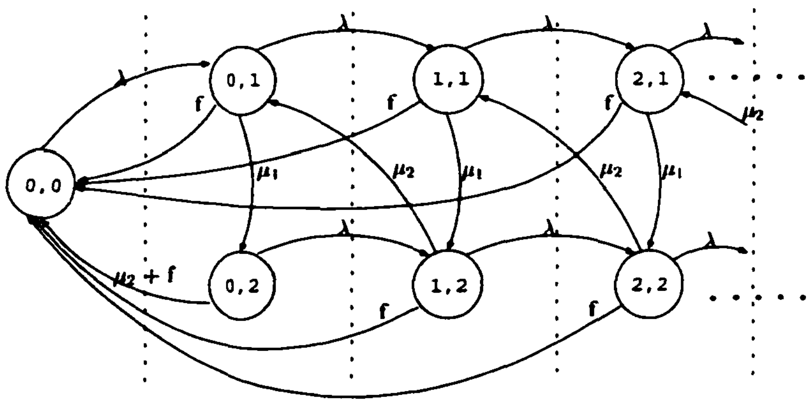

Figure 2.3: The state transition diagram of a GI/M/1 process with failures.

process is

$$\mathbf{Q}_{GI/Hypo2/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(2)} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(3)} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{2.19}$$

where the matrices $\widehat{\mathbf{L}}, \widehat{\mathbf{F}}, \mathbf{F}, \mathbf{L}, \mathbf{B}$ and $\mathbf{B}^{(j)}$, $j \geq 1$ are defined as follows:

$$\mathbf{F} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} -\alpha_1 & \mu_1 \\ 0 & -\alpha_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ \mu_2 & 0 \end{bmatrix}, \quad \widehat{\mathbf{L}} = \begin{bmatrix} -\lambda' & \lambda' & 0 \\ f & -\alpha_1 & \mu_1 \\ \mu_2 + f & 0 & -\alpha_2 \end{bmatrix},$$

$$\widehat{\mathbf{F}} = \begin{bmatrix} 0 & 0 \\ \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \quad \widehat{\mathbf{B}}^{(1)} = \begin{bmatrix} f & 0 & 0 \\ f & \mu_2 & 0 \end{bmatrix} \quad \widehat{\mathbf{B}}^{(j)} = \begin{bmatrix} f & 0 & 0 \\ f & 0 & 0 \end{bmatrix}, \quad j \geq 2. \tag{2.20}$$

where $\alpha_j \overset{\text{def}}{=} \lambda + \mu_j + f$, $j = 1, 2$.

The process with state transition diagram in Figure 2.3 is an example of a GI/M/1-type or *skip-free* to the right process. The block partitioned infinitesimal generator $\mathbf{Q}_{GI/M/1}$ of a GI/M/1-type process resembles the infinitesimal generator of the GI/M/1 queue:

$$\mathbf{Q}_{GI/M/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.21}$$

$\mathbf{Q}_{GI/M/1}$ is a *lower Hessenberg* type matrix, i.e., its blocks above the main diagonal, but the first one, are all zero matrices. As illustrated in Figure 2.3, examples of GI/M/1 type processes include systems that allow the jobs to be served in bulks and systems that capture failure of service nodes [34]. We elaborate on solution techniques for GI/M/1-type process in Chapter 3.

2.5.3 M/G/1-type processes

Consider again the example of the M/Hypo(2)/1 queue. Now suppose that the arrivals can occur in bulk, i.e., one job arrives with rate λ, two jobs arrive *simultaneously* with rate $\lambda/2$, three jobs arrive simultaneously with rate $\lambda/4$, i.e., the bulk sizes decrease geometrically. The service process consists of two exponential phases in series. The state transition diagram of this process is illustrated in Figure 2.4.

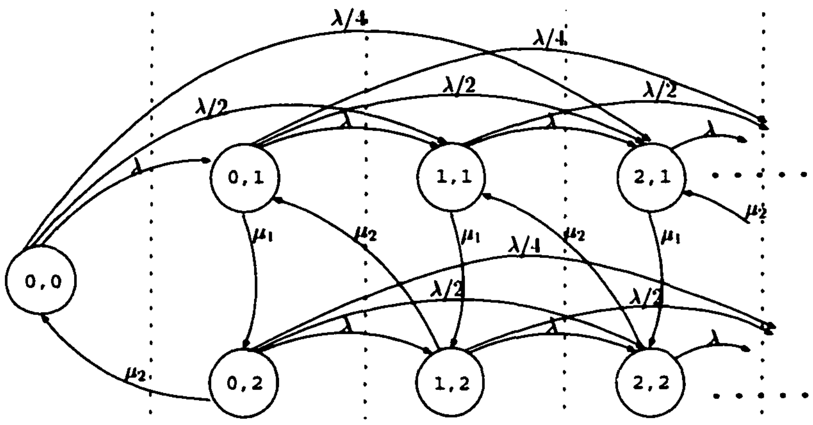

Figure 2.4: The state transition diagram of a M/Hypo(2)1 queue with bulk arrivals.

The processes with similar patterns in the embedded Markov chain are known as M/G/1-type or *skip-free* to the left processes. Their infinitesimal generator matrix

$\mathbf{Q}_{M/G/1}$ can be block-partitioned as:

$$\mathbf{Q}_{M/G/1} = \begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}}^{(1)} & \hat{\mathbf{F}}^{(2)} & \hat{\mathbf{F}}^{(3)} & \hat{\mathbf{F}}^{(4)} & \cdots \\ \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.22}$$

The infinitesimal generator $\mathbf{Q}_{M/G/1}$ of an M/G/1 type process is an *upper Hessenberg* matrix, i.e., the blocks below the main block-diagonal, but the first one, are all zero matrices. As depicted in Figure 2.4, M/G/1-type processes usually characterize bulk arrivals, i.e., more than one job may arrive in the queueing system at a time [34]. We give details on solution methods for M/G/1-type processes in Chapter 3.

2.5.4 GI/G/1-type processes

Now consider a queueing system where both arrivals and service may occur in bulk. Using a similar example as the ones in Figures 2.4 and 2.3, the state transition diagram of the embedded Markov chain of such process is illustrated in Figure 2.5.

The processes with similar patterns in the embedded Markov chain are known as GI/G/1-type processes and are generalizations of the M/G/1 and GI/M/1-type processes. Their infinitesimal generator matrix $\mathbf{Q}_{GI/G/1}$ can be block-partitioned as:

$$\mathbf{Q}_{GI/G/1} = \begin{bmatrix} \mathbf{L}^{(0)} & \hat{\mathbf{F}}^{(1)} & \hat{\mathbf{F}}^{(2)} & \hat{\mathbf{F}}^{(3)} & \hat{\mathbf{F}}^{(4)} & \cdots \\ \hat{\mathbf{B}}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \hat{\mathbf{B}}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \hat{\mathbf{B}}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \hat{\mathbf{B}}^{(4)} & \mathbf{B}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.23}$$

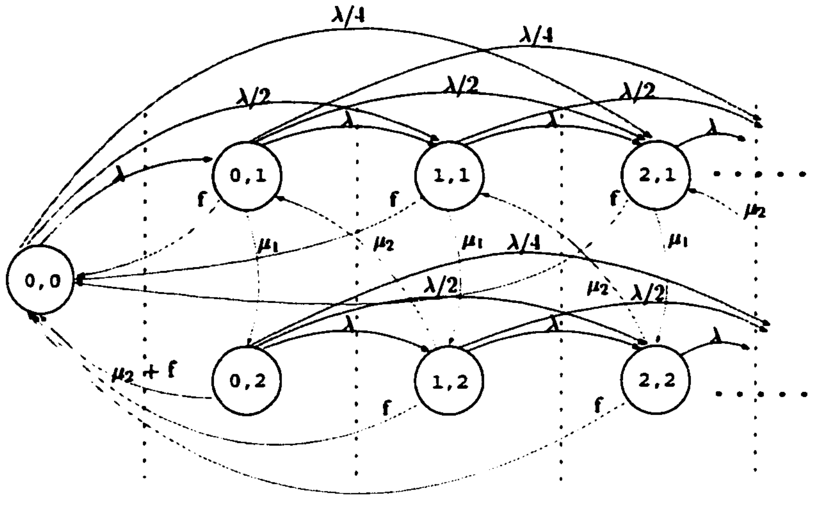

Figure 2.5: The state transition diagram of a GI/G/1 queue.

The infinitesimal generator $Q_{GI/G/1}$ of a GI/G/1 type process is a *full* matrix. However the matrix is structured in repeating blocks. As depicted in Figure 2.5, GI/G/1-type processes usually characterize systems with bulk arrivals and possible failures.

2.6 Phase-type distributions

Phase-type (PH) distributions are based on the method of stages technique that was introduced by Erlang and generalized by Neuts [67]. PH distributions consist of a "general" mixture of exponentials and are characterized by a *finite* and *absorbing* Markov chain. The number n of phases in the PH distribution is equal to the number of transient states in the associated (underlying) Markov chain. A PH distribution represents random variables that are measured by the time X that the underlying

Markov chain spends in its transient portion till absorption. From this perspective, a row vector $\boldsymbol{\tau}$ of size n is associated with the underlying Markov chain of a PH distribution and represents the initial probability vector for each of its transient states. The infinitesimal generator of the underlying Markov chain of a PH distribution is shown in Eq.(2.24). The zero row represent the absorbing state of the chain.

$$\mathbf{Q} = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{t} & \mathbf{T} \end{bmatrix} \tag{2.24}$$

$\mathbf{T}$ is an $n \times n$ matrix representing the transitions among the transient states and $\mathbf{t}$ is a column vector of size n representing the transitions from the transient states to the absorbing state of the underlying Markov chain. Matrix $\mathbf{T}$ and vector $\mathbf{t}$ relate to each other as $\mathbf{t} = -\mathbf{T} \cdot \mathbf{e}$. The PH distribution is fully represented by the vector $\boldsymbol{\tau}$ and the matrix $\mathbf{T}$. The basic characteristics of a PH distribution are

- the cumulative distribution function: $F(x) = 1 - \boldsymbol{\tau} e^{\mathbf{T}x}\mathbf{e}$,
- the density function: $f(x) = \boldsymbol{\tau} e^{\mathbf{T}x}(-\mathbf{T} \cdot \mathbf{e})$,
- the n^th moment: $m_n = (-1)^n \cdot n! \boldsymbol{\tau} \mathbf{T}^{-n}\mathbf{e}$,

where $\mathbf{e}$ is a column vector of ones with the appropriate dimension. Observe that in the PDF and CDF of a PH distribution, the matrix $\mathbf{T}$ appears as an exponential coefficient. As such, PH distributions are called matrix-exponential distributions. The term $e^{\mathbf{T}x}$ is defined as

$$e^{\mathbf{T}x} = \sum_{n \geq 0} \frac{1}{n!}(\mathbf{T}x)^n.$$

Obviously the number of parameters that define a PH distribution is $n(n+1)$, however the cases of PH distributions that are used in practice have the majority of the values in $\boldsymbol{\tau}$ and $\mathbf{T}$ equal to zero. For the hyperexponential, hypoexponential, and Coxian[6] distributions, special cases of PH distributions, the definitions of $\boldsymbol{\tau}$ and $\mathbf{T}$ are as follows (assuming $n = 4$).

	Hypoexponential	Coxian	Hyperexponential
$\boldsymbol{\tau}$	$[1, 0, 0, 0]$	$[1, 0, 0, 0]$	$[p_1, p_2, p_3, p_4]$
$\mathbf{T}$	$\begin{bmatrix} -\lambda_1 & \lambda_1 & 0 & 0 \\ 0 & -\lambda_2 & \lambda_2 & 0 \\ 0 & 0 & -\lambda_3 & \lambda_3 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix}$	$\begin{bmatrix} -\lambda_1 & \lambda_1^1 & 0 & 0 \\ 0 & -\lambda_2 & \lambda_2^1 & 0 \\ 0 & 0 & -\lambda_3 & \lambda_3^1 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix}$	$\begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & -\lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix}$

Table 2.1: Structure of $\boldsymbol{\tau}$ and $\mathbf{T}$ for the Hypoexponential, Coxian, and Hyperexponential distributions.

Since the structure of the underlying Markov chain is arbitrary, a PH distribution captures a wide range of characteristics including high variability. The PH random variables are independently identically distributed because the initial probability vector $\boldsymbol{\tau}$ is the same every time the Markov chain starts in its transient portion in the corresponding renewal process. A PH distribution with infinite number of phases captures exactly a power-tailed distribution [71]. There is another set of processes known as Markovian Arrival Processes (MAPs) that are still based on the method of exponential stages and capture complex characteristics such as short- and long-range dependence. PH distributions are a special case of MAPs [69].

[6]A given exponential phase i, $0 < i < n$, of a Coxian distribution is associated with two rates: λ_i^1 for reaching the next exponential phase and λ_i^2 for reaching the absorbing state of the underlying Markov chain. The total rate of leaving phase i is $\lambda_i = \lambda_i^1 + \lambda_i^2$.

2.6.1 Fitting algorithms for PH distributions

The fitting techniques available for PH distributions are either based on moment matching or on maximum likelihood estimators (MLEs) (see [48] for more details in MLEs). Moment matching techniques are computationally efficient, but apply to somewhat more restrictive models [42]. Among the techniques that are based on MLEs, we distinguish the Expectation-Maximization (EM) algorithm for fitting both data and distributions into general PH distributions [6], MLAPH [13] which uses MLEs to fit continuous distribution functions into acyclic PH distributions, and a numerical optimization method to fit long-tailed distribution functions into Coxian distributions [40]. The Feldmann-Whitt (FW) algorithm [30] is a heuristic-based approach to fit long-tailed distribution functions such as Weibull and Pareto into hyperexponential distributions. In the following we discuss in more details the EM and FW fitting algorithms.

2.6.1.1 EM Algorithm

The Expectation-Maximization (EM) algorithm is an iterative method that aims to maximize the log-likelihood function of a data set. In the case of PH distributions, the likelihood function is given by the formula

$$L(\boldsymbol{\tau}, \mathbf{T}; \mathbf{y}) = \prod_{i=1}^{n} \boldsymbol{\tau} e^{\mathbf{T} y_i}(-\mathbf{T}\mathbf{e}). \tag{2.25}$$

where $\mathbf{y}$ represents the data set to be fitted into PH distribution. The EM algorithm starts from an initial guess for the parameters $(\boldsymbol{\tau}^{(0)}, \mathbf{T}^{(0)})$. Each iteration j generates a new estimate $(\boldsymbol{\tau}^{(j)}, \mathbf{T}^{(j)})$, such that the value of the likelihood function with the new estimate is larger than the previous ones

$$L(\boldsymbol{\tau}^{(j)}, \mathbf{T}^{(j)}; \mathbf{y}) \leq L(\boldsymbol{\tau}^{(j+1)}, \mathbf{T}^{(j+1)}; \mathbf{y}).$$

The sequence of the estimates $(\boldsymbol{\tau}^{(0)}, \mathbf{T}^{(0)}), (\boldsymbol{\tau}^{(1)}, \mathbf{T}^{(1)}), \ldots, (\boldsymbol{\tau}^{(N)}, \mathbf{T}^{(N)})$ should eventually converge after N steps to the stationary point $(\hat{\boldsymbol{\tau}}, \hat{\mathbf{T}})$ of the likelihood. This stationary point is hopefully the one to maximize the likelihood. however there are no guarantees, as the algorithm can get stuck in insignificant local maximums or saddle points. The best accuracy in the fittings is achieved when different initial values are tried. More details on the EM algorithm for fitting data sets into PH distributions as well as fitting any non-negative continuous distribution into a PH distribution can be found in [6].

EM is one of the best algorithms to approximate data sets or continuous distributions by other continuous distributions. For the case of fitting data sets with long-tails into PH distributions, EM may not capture the tail correctly since the procedure searches for a global maximum [40]. As an iterative procedure, its computation complexity depends on the size of the data set and the number of unique entries, as well as the number n of phases for the fitted PH distribution.

2.6.1.2 Feldmann-Whitt (FW) Algorithm

The Feldmann-Whitt (FW) algorithm [30] is a recursive heuristic that fits complete monotone non-negative continuous distributions into hyperexponential distributions. The idea behind the algorithm is to first fit the rightmost portion of the tail into a single exponential distribution determining its parameter and the associated probability (see Table 2.1 for the definition of hyperexponential distribution). Once the first pair of parameters is known the weight of the fitted portion is subtracted from the distribution (FW deals with the complementary cumulative distribution function) and again the rightmost tail of the result is fitted into another exponential distribution by computing the same set of parameters. The algorithm continues until the parameters of all phases of the hyperexponential distribution are computed.

The FW algorithm is an elegant and accurate fitting approach for distributions like Pareto and Weibull (for a specific range of parameters), that are characterized by high variability. Nevertheless, the FW fits are less accurate when the PDF of the distribution is not completely monotone, like in Lognormal or Weibull (for a specific range of parameters) distributions. Another drawback of the FW algorithm is that it can apply to distribution functions only and not to data sets. Hence, an additional step is required in the fitting procedure, i.e., to first fit the data set into a Weibull or Pareto, before using the FW algorithm. This step may introduces additional errors to the final fit. The FW algorithm is outlined in detail in Appendix A.

2.7 Markovian Arrival Process

The Markovian Arrival Process (MAP), introduced by Neuts in [69], is a generalization of the PH distribution. A MAP is associated with a finite absorbing Markov chain but instead of having only one initial probability vector $\boldsymbol{\tau}$, it requires as many as the number of transient states in the underlying Markov chain, i.e., one $\boldsymbol{\tau}$ per transient state. This means that once the Markov chain has entered the absorbing state and a single MAP random variable is generated, the process restarts from the transient part again for the next random variable "remembering" the last transient state that reached absorption. Therefore, the MAP is formally represented by two matrices $(\mathbf{D}_0, \mathbf{D}_1)$ rather than a matrix and a vector as in the PH-distribution case and is in general a non-renewal process. Matrix $\mathbf{D}_0$, similarly to matrix $\mathbf{T}$ for a PH distribution, describes the interaction between the transient states of the underlying Markov chain. All off-diagonal entries of $\mathbf{D}_0$ are non-negative. Matrix $\mathbf{D}_1$ describes how the transient states of the underlying Markov chain are re-entered once the absorption is reached. Matrix $\mathbf{D}_1$ has only non-negative entries. We note that $\mathbf{D}_0 + \mathbf{D}_1$ is an infinitesimal generator describing the transitions of the transient states in the underlying Markov chain.

The PH distribution is a MAP where matrix $\mathbf{D}_1$ has equal rows. The most popular case of a MAP is known as Markov Modulated Poisson Process (MMPP) where the matrix $\mathbf{D}_1$ has all entries zero except the diagonal ones. Even a Poisson process is a MAP with only one single transient state in the underlying Markov chain.

The concept of the Markovian Arrival Process is further extended to allow for simultaneous batch absorptions in the underlying Markov chain. The resulting process is known as the *Batch Markovian Arrival Process* (BMAP) [54]. Formally a BMAP is represented by an infinite number of matrices $\mathbf{D}_i$ for $i \geq 0$ allowing for an arbitrary large batch size. Similarly to a MAP, $\sum_{i=0}^{\infty} \mathbf{D}_i$ is an infinitesimal generator. The importance of (B)MAPs lies in their ability to be more effective and powerful traffic models than the simple Poisson process or the batch Poisson process, as they can effectively capture dependence and correlation, salient characteristics of the arrival process in Internet-related systems [70, 38]. Although in this dissertation we focus on systems with dependent structure only in the arrival process, we stress that the both the arrival and service processes in a queueing system can exhibit long-range dependence. Similar to a PH distribution with infinite number of phases capturing exactly the power-tailed behavior of a distribution, a MAP with infinite number of states captures exactly the long-range behavior of a stochastic process.

In our exposition, we measure the dependent behavior in a stochastic process using the *Hurst parameter*, H, as a metric. If the Hurst parameter is larger than 0.5 then the stochastic process exhibits long-range dependence. The long-range dependent behavior in a stochastic process is stronger as the value of the Hurst parameter increases. For rigorous definitions and discussions on short- and long-range dependent processes please refer to [10].

2.7.1 Fitting techniques for MAPs

As in the case of PH distributions, moment matching and MLE-based techniques exists for fitting data sets into MAPs. The moment matching methods tend to be computationally more efficient, but the models are somewhat restrictive with respect to the structure of the underlying Markov chain. The moment matching techniques for MAPs mainly deal with 2 state MMPPs [89, 35].

One of the early MLE-based techniques for fitting data sets into a 2-state MMPPs is proposed by Meier-Hellstern [56]. Optimization methods have been proposed by Ryden [92] as part of an MLE-based approach for MMPP parameter estimation. Horvath et al. [39] develop a heuristic fitting method for MAPs based on the superposition of phase-type and interrupted Poisson processes. Also the EM algorithm is used as an estimation procedure for MAPs and their more general form, BMAPs [15, 16].

2.8 Aggregation techniques

For the analysis of complex Markov chains with large state spaces, ***decomposition and aggregation*** techniques are used as tools that reduce the complexity of the solution. The general idea behind decomposition and aggregation is to decompose (divide) the Markov chain in ***independent*** subchains which can be solved separately, and then aggregate (combine) the results back together. If a Markov chain can be divided into independent subchains then this Markov chain is ***completely decomposable***. In practice, it is unusual to find completely decomposable Markov chains. However, there are cases where the condition of independent subchains *almost* holds. These type of Markov chains are called *nearly completely decomposable.*

A Markov chain is nearly completely decomposable if its state space S can be partitioned into subsets such that within the same subset, the states interact *strongly* with one another, i.e., are strongly connected, while between different subsets the states interact loosely with one another, i.e., are loosely connected. For a CTMC, this means that the cumulative rate between states of the same subset is much higher than the cumulative rate between the states of different subsets [27]. The fundamental formal foundation of decomposition methods for the steady state analysis of queueing systems is due to Simon and Ando [96].

Generally speaking, aggregation and decomposition techniques yield approximate solutions. If they apply to the nearly completely decomposable Markov chains, the relative error can be maintained under a given small constant. If the conditions of nearly decomposable Markov chains do not hold, then the error introduced in the results is considerably large, with the notable exception of product-form queueing networks where the aggregation techniques yield exact solutions. The value of the decomposition and aggregation methodology is two-fold [26]:

- it provides efficient numerical solutions for complex and large Markov chains by dividing the problem into smaller and more tractable subproblems, and
- provides theoretical insights regarding the structure of the process modeled by

the Markov chain.

2.8.1 Exact aggregation and decomposition

The basic idea behind the aggregation/decomposition is to group states in strongly connected subchains that are loosely connected to each other and:

(a) analyze and solve each subchain as if there is no interaction among them,

(b) analyze the interaction among subchains, ignoring the interaction within each subchain.

The infinitesimal generator matrix $\mathbf{Q}_{NCD}$ of nearly decomposable Markov chains should be partitioned in such a way that the off-diagonal elements are *very* small compared to the elements of the diagonal blocks

$$\mathbf{Q}_{NCD} = \begin{bmatrix} \mathbf{Q}_{0,0} & \mathbf{Q}_{0,1} & \cdots & \mathbf{Q}_{0,N-1} \\ \mathbf{Q}_{1,0} & \mathbf{Q}_{1,1} & \cdots & \mathbf{Q}_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{N-1,0} & \mathbf{Q}_{N-1,1} & \cdots & \mathbf{Q}_{N-1,N-1} \end{bmatrix}. \tag{2.26}$$

The diagonal blocks $\mathbf{Q}_{i,i}$, for $i = 0, \ldots, N-1$, are square with dimension n_i $i = 0, \ldots, N-1$, where $\sum_{i=0}^{N-1} n_i = n$, is the cardinality of the state space $\mathcal{S}$.

The solution follows the steps outlined at the beginning of this subsection. First, we assume that $\mathbf{Q}_{NCD}$ is completely decomposable, that is, the off-diagonals blocks in Eq.(2.26) are $\mathbf{0}$. We solve the N different Markov chains defined by the diagonal blocks $\mathbf{Q}_{i,i}$, for $i = 0, \ldots, N-1$ in Eq.(2.26). If any of these diagonal blocks is not an infinitesimal generator itself, which is usually the case when $\mathbf{Q}_{NCD}$ is nearly completely decomposable, the sum of the probabilities in the off-diagonal blocks is used as a correction factor to the elements of the diagonal blocks of $\mathbf{Q}_{NCD}$. This correction process affects the accuracy of the solution.

Let $\boldsymbol{\alpha}^{(i)}$ be the stationary distribution vector obtained by solving the subchain with infinitesimal generator matrix $\mathbf{Q}^*_{i,i}$[7] for $i = 0, ..., N-1$. The element $\boldsymbol{\alpha}^{(i)}[j]$ for $i = 0, ..., N-1$ and $j = 0, ..., n_i - 1$ is the probability of being in state j of subchain i, given that the process finds itself in subchain i. We need to compute $||\mathbf{a}[i]||$ for $i = 0, ..., N-1$, which are the probabilities of being in subchain i for $i = 0, ..., N-1$, such that we can eliminate the condition factor from the elements of the probability vectors $\boldsymbol{\alpha}^{(i)}$, to construct the stationary probability vector $\boldsymbol{\pi}$ of the original process with infinitesimal generator matrix $\mathbf{Q}_{NCD}$. We define an infinitesimal generator matrix $\mathbf{Q}_c$ with $N \times N$ elements as

$$\mathbf{Q}_c = \begin{bmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,N-1} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N-1,0} & q_{N-1,1} & \cdots & q_{N-1,N-1} \end{bmatrix}. \tag{2.27}$$

where $q_{i,j} = \boldsymbol{\alpha}^{(i)}\mathbf{Q}_{i,j}\mathbf{1}^T$ for $i \neq j, i, j = 0, ..., N-1$. The infinitesimal generator matrix $\mathbf{Q}_c$ represent a new irreducible CTMC whose stationary distribution $\mathbf{a}$ can be computed based on Eq.(2.4.2) [107]. After computing $\mathbf{a}$, the stationary distribution $\boldsymbol{\pi}$ of the original process is constructed as $\boldsymbol{\pi} = [\mathbf{a}[0]\boldsymbol{\alpha}^{(0)}, ..., \mathbf{a}[N-1]\boldsymbol{\alpha}^{(N-1)}]$. The

[7] $\mathbf{Q}^*_{i,i}$ means that the respective matrix $\mathbf{Q}_{i,i}$ is modified by adding the corresponding off-diagonal cumulative rate into its elements.

infinitesimal generator $\mathbf{Q}_c$ is known as the *coupling matrix*. More details on the coupling matrix and the *coupling process* are given in the following subsection.

2.8.2 Stochastic complementation

In this subsection, we outline the concept of stochastic complementation [61] which is another aggregation technique. Although there may be similarities with the technique presented in Subsection 2.8.1, the methodology is different. The concept of stochastic complementation is formally defined for finite state spaces [61]. Here, we define it for the infinite case, which is a straightforward extension. Since throughout this dissertation we consider continuous time Markov chains, we define and state the concept of stochastic complementation in terms of CTMCs but they readily apply to DTMCs as well (via uniformization [67]).

Let partition the state space $\mathcal{S}$ of an ergodic CTMC with infinitesimal generator matrix $\mathbf{Q}$ and stationary probability vector $\boldsymbol{\pi}$, satisfying $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$, into two disjoint subsets, $\mathcal{A}$ and $\overline{\mathcal{A}}$.

Definition [61] (Stochastic complement) The stochastic complement of $\mathcal{A}$ is

$$\overline{\mathbf{Q}} = \mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}](-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}], \tag{2.28}$$

where $(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}[r,c]$ represents the mean time spent in state $c \in \overline{\mathcal{A}}$, starting from state $r \in \overline{\mathcal{A}}$, before reaching any state in $\mathcal{A}$, and $((-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}])[r,c']$

represents the probability that, starting from $r \in \overline{\mathcal{A}}$, the process enters $\mathcal{A}$ through state c'. □

The stochastic complement $\overline{\mathbf{Q}}$ is the infinitesimal generator of a new CTMC which mimics the original CTMC but "skips over" states in $\overline{\mathcal{A}}$. The following theorems formalizes this concept.

Theorem [61] The stochastic complement $\overline{\mathbf{Q}}$ of $\mathcal{A}$ is an infinitesimal generator matrix and it is irreducible if $\mathbf{Q}$ is. □

Theorem [61] If $\boldsymbol{\alpha}$ is the stationary probability vector of $\overline{\mathbf{Q}}$, satisfying $\boldsymbol{\alpha} \cdot \overline{\mathbf{Q}} = \mathbf{0}$,

$$\boldsymbol{\alpha} = \frac{\boldsymbol{\pi}[\mathcal{A}]}{\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T} = Norm(\boldsymbol{\pi}[\mathcal{A}]). \tag{2.29}$$

□

In other words, $\boldsymbol{\alpha}$ gives the individual probability of being in each state of $\mathcal{A}$ for the original CTMC, conditioned on being in $\mathcal{A}$. This implies that the stationary probability distribution $\boldsymbol{\alpha}$ of the stochastic complement differs from the corresponding portion of the stationary distribution vector of the original CTMC, $\boldsymbol{\pi}[\mathcal{A}]$, only by a constant $\alpha = \boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T$, which represents the probability of being in $\mathcal{A}$ in the original CTMC. The value α is also known as the *coupling factor* of the stochastic complement of states in $\mathcal{A}$.

Assume that we know $\boldsymbol{\alpha}$ and $\overline{\boldsymbol{\alpha}}$, the stationary probability vectors of the stochastic complements of states in $\mathcal{A}$ and $\overline{\mathcal{A}}$, respectively. The stationary probability vector of the original CTMC is given by

$$\boldsymbol{\pi} = [\alpha \cdot \boldsymbol{\alpha}, \overline{\alpha} \cdot \overline{\boldsymbol{\alpha}}], \tag{2.30}$$

where α and $\overline{\alpha}$ are the two respective coupling factors. The coupling factors must sum to one and the following theorem holds.

Theorem [61] (Coupling) The coupling factors α and $\overline{\alpha}$ are the stationary probabilities for a new 2-state CTMC with infinitesimal generator the coupling matrix $\mathbf{Q}_c$, where $\mathbf{Q}_c[0,1] = \mathbf{Q}_c[\mathcal{A}, \overline{\mathcal{A}}] = \boldsymbol{\alpha} \cdot \mathbf{Q}[\mathcal{A}, \overline{\mathcal{A}}] \cdot \mathbf{1}^T$ and $\mathbf{Q}_c[1,0] = \mathbf{Q}_c[\overline{\mathcal{A}}, \mathcal{A}] = \overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}] \cdot \mathbf{1}^T$. □

In Subsection 6.1, we present more results on the coupling process. Differently from the above theorem, we extend the coupling process by allowing subsets $\mathcal{A}$ and $\overline{\mathcal{A}}$ to be non-disjoint. Furthermore in Subsection 6.2, we present *pseudo stochastic complementation*, which is another way of implementing stochastic complementation.

There are cases where we can take advantage of the special structure of the CTMC and explicitly generate the stochastic complement of states in $\mathcal{A}$. To consider these cases, we rewrite the definition of stochastic complement in Eq.(2.28) as

$$\overline{\mathbf{Q}} = \mathbf{Q}[\mathcal{A}, \mathcal{A}] + \mathit{RowSum}(\mathbf{Q}[\mathcal{A}, \overline{\mathcal{A}}])\mathbf{Z}, \tag{2.31}$$

where $\mathbf{Z} = \mathit{Norm}(\mathbf{Q}[\mathcal{A}, \overline{\mathcal{A}}])\ (-\mathbf{Q}[\overline{\mathcal{A}}, \overline{\mathcal{A}}])^{-1}\mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}]$. The r^{th} row of $\mathbf{Z}$, which sums to one, specifies how this rate should be redistributed over the states in $\mathcal{A}$ when the process eventually reenters it, while the r^{th} diagonal element of $RowSum(\mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}])$ represents the rate at which the set $\mathcal{A}$ is left from its r^{th} state to reach any of the states in $\overline{\mathcal{A}}$.

Lemma (Single entry) If $\mathcal{A}$ can be entered from $\overline{\mathcal{A}}$ only through a single state $c \in \mathcal{A}$, the matrix $\mathbf{Z}$ defined in Eq. (2.31) is trivially computable: it is a matrix of zeros except for its c^{th} column, which contains all ones. □

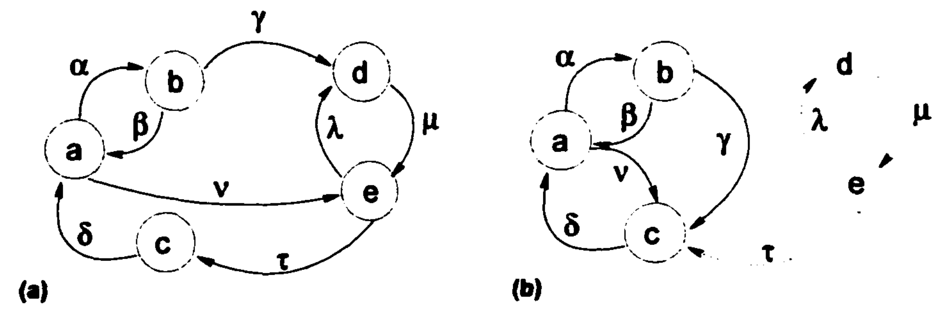

Figure 2.6: Stochastic Complementation for a finite Markov chain.

We choose the simple finite Markov chain depicted in Figure 2.6(a) to explain the concept of stochastic complementation. The state space of this Markov chain is $\mathcal{S} = \{a, b, c, d, e\}$. We construct the stochastic complement of the states in set $\mathcal{A} = \{a, b, c\}$ ($\overline{\mathcal{A}} = \{d, e\}$), as shown in Figure 2.6(b). The matrices used in Eq.(2.28) for this example are:

$$\mathbf{Q}[\mathcal{A},\mathcal{A}] = \begin{bmatrix} -(\alpha+\nu) & \alpha & 0 \\ \beta & -(\gamma+\beta) & 0 \\ \delta & 0 & -\delta \end{bmatrix}, \qquad \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] = \begin{bmatrix} 0 & \nu \\ \gamma & 0 \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \tau \end{bmatrix}, \qquad \mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}] = \begin{bmatrix} -\mu & \mu \\ \lambda & -(\lambda+\tau) \end{bmatrix}.$$

Observe that in this case $\mathcal{A}$ is entered from states in $\overline{\mathcal{A}}$ only through state c and the stochastic complement can be trivially constructed based on the above lemma on single entry. There are only two transitions from states in $\mathcal{A}$ to states in $\overline{\mathcal{A}}$; the transition with rate γ from state b to state d and the transition with rate ν from state a to state e. These two transitions are folded back into $\mathcal{A}$ through state c, which is the single entry in $\mathcal{A}$. The following derivation shows that because of the special single entry state the two folded transitions have the original rates. γ and ν respectively.

$$\mathbf{Z} = Norm(\mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}])(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{\lambda+\tau}{\mu\tau} & \frac{1}{\tau} \\ \frac{\lambda}{\mu\tau} & \frac{1}{\tau} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \tau \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

which further results in:

$$RowSum(\mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}]) \cdot \mathbf{Z} = \begin{bmatrix} \nu & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \nu \\ 0 & 0 & \gamma \\ 0 & 0 & 0 \end{bmatrix}.$$

2.9 Chapter summary

In this chapter, we gave an overview of basic concepts and notations. We focused on Markov chains, their definition, classification, and solution methods. We described aggregation/decomposition solution techniques for Markov chains, elaborating in detail on the stochastic complementation, a technique that we often refer throughout this dissertation. We illustrated via simple examples the structure of infinite Markov chains with repetitive structures. Finally, we described tractable stochastic processes such as PH distributions and MAPs, which we later use as inputs to queueing models.

Chapter 3

Matrix-Analytic Methods

Matrix analytic techniques provide a framework that is widely used for the exact analysis of a general and frequently encountered class of queueing models. In these models, the embedded Markov chains are two-dimensional generalizations of elementary GI/M/1 and M/G/1 queues [43], and their intersection, i.e., quasi-birth-death (QBD) processes. GI/M/1 and M/G/1 Markov chains model systems with inter-arrival and service times characterized, respectively, by *general* distributions rather than simple exponentials and are often used as the modeling tool of choice in modern computer and communication systems [65, 78, 101, 29, 53]. Alternatively, GI/M/1 and M/G/1 Markov chains can be analyzed by means of eigenvalues and eigenvectors [32]. The class of models that can be analyzed using M/G/1-type Markov chains includes the important class of BMAP/G/1 queues, where the arrival process and/or service process can exhibit correlation and long-tail, i.e., salient characteristics of Internet-related systems.

Neuts [67] defines various classes of infinite-state Markov chains with a repetitive structure, whose state space is partitioned into the boundary states $\mathcal{S}^{(0)} = \{s_1^{(0)}, \ldots, s_m^{(0)}\}$ and the sets of states $\mathcal{S}^{(i)} = \{s_1^{(i)}, \ldots, s_n^{(i)}\}$, for $i \geq 1$, that correspond to the repetitive portion of the chain. In Section 2.5, we identified these classes as GI/M/1-type, M/G/1-type, and their intersection, QBD processes. The structure of the infinitesimal generator for these type of processes is shown in Eqs.(2.21), (2.22), and (2.18), respectively.

For systems of the M/G/1-type, matrix analytic methods have been proposed for the solution of the basic equation $\boldsymbol{\pi} \cdot \mathbf{Q}_{M/G/1} = \mathbf{0}$ [69]. Key to the matrix-analytic methods is the computation of an auxiliary matrix called **G**. Traditional solution methodologies for M/G/1-type processes compute the stationary probability vector with a recursive function based on **G**. Iterative algorithms are used to determine **G** [60, 47].

The solution of GI/M/1-type processes is significantly simpler than the solution of M/G/1-type processes because of the *matrix geometric* relation [67] that exists among the stationary probabilities of sets $\mathcal{S}^{(i)}$ for $i \geq 1$. This property leads to significant algebraic simplifications resulting in the very elegant matrix-geometric solution technique. Key to the matrix-geometric solution is a matrix called **R** which is used in the computation of the steady-state probability vector and measures of interest.

Since QBDs are special cases of both M/G/1-type processes and GI/M/1-type processes, either the matrix-analytic method or the matrix-geometric solution can be used for their analysis. The matrix-geometric solution is the preferable one because of its simplicity. Both matrices $\mathbf{G}$ and $\mathbf{R}$ are defined for QBD processes.

Key to the solution of Markov chains of the M/G/1, GI/M/1, and QBD types, is the existence of the repetitive structure, as illustrated in Eqs. (2.22), (2.21), and (2.18). This special structure allows for a certain recursive procedure to be applied for the computation of the stationary probability vector $\boldsymbol{\pi}^{(i)}$ corresponding to $\mathcal{S}^{(i)}$ for $i \geq 1$. It is this recursive relation that gives elegance to the solution for the case of GI/M/1 (and consequently QBD) Markov chains, but results in unfortunately more complicated mathematics for the case of the M/G/1-type.

3.1 Matrix geometric solutions for GI/M/1-type and QBD processes

In this section, we give a brief overview[1] of the matrix geometric solution technique for GI/M/1-type and QBD processes. While QBDs fall under both the M/G/1 and the GI/M/1-type cases, they are most commonly associated with GI/M/1 processes because they can be both solved using the well-known matrix geometric approach [67].

Key to the general solution for the generator of Eqs.(2.21) and (2.18) is the as-

[1]In this section and in the remainder of this dissertation, we assume continuous time Markov chains, or CTMCs unless otherwise stated, but our discussion applies just as well to discrete time Markov chains, or DTMCs.

sumption that a geometric relation[2] holds among the stationary probability vectors $\boldsymbol{\pi}^{(i)}$ of states in $\mathcal{S}^{(i)}$ as follows:

$$\boldsymbol{\pi}^{(i)} = \boldsymbol{\pi}^{(1)} \cdot \mathbf{R}^{i-1} \quad \forall i \geq 1, \tag{3.1}$$

where, in the GI/M/1-type case. $\mathbf{R}$ is the solution of the matrix equation

$$\mathbf{F} + \mathbf{R} \cdot \mathbf{L} + \sum_{k=1}^{\infty} \mathbf{R}^{k+1} \cdot \mathbf{B}^{(k)} = \mathbf{0}, \tag{3.2}$$

and can be computed using iterative numerical algorithms. The above equation is obtained from the flow balance equations of the repeating portion of the process. i.e.. starting from the third column of $\mathbf{Q}_{GI/M/1}$. One can solve a GI/M/1-type process starting from the flow balance equations corresponding to the first two columns of $\mathbf{Q}_{GI/M/1}$. By substituting $\boldsymbol{\pi}^{(i)}$ for $i \geq 2$ with their equivalents from Eq. (3.1), i.e.. $\boldsymbol{\pi}^{(1)}\mathbf{R}^{i-1}$, and adding the normalization condition as

$$\boldsymbol{\pi}^{(0)} \cdot \mathbf{1}^T + \boldsymbol{\pi}^{(1)} \cdot \sum_{i=1}^{\infty} \mathbf{R}^{i-1} \cdot \mathbf{1}^T = 1 \quad \text{i.e.,} \quad \boldsymbol{\pi}^{(0)} \cdot \mathbf{1}^T + \boldsymbol{\pi}^{(1)} \cdot (\mathbf{I} - \mathbf{R})^{-1} \cdot \mathbf{1}^T = 1.$$

we obtain the following system of linear equations

$$[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}] \cdot \begin{bmatrix} \mathbf{e} & (\mathbf{L}^{(0)})^{\circ} & \widehat{\mathbf{F}}^{(1)} \\ (\mathbf{I} - \mathbf{R})^{-1} \cdot \mathbf{e} & \left(\sum_{k=1}^{\infty} \mathbf{R}^{k-1} \cdot \widehat{\mathbf{B}}^{(k)}\right)^{\circ} & \mathbf{L} + \sum_{k=1}^{\infty} \mathbf{R}^{k} \cdot \mathbf{B}^{(k)} \end{bmatrix} = [1, \mathbf{0}], \tag{3.3}$$

[2]This is similar to the simplest degenerate case of a QBD process. the straight forward birth-death M/M/1 case.

that yields a unique solution for $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$. The symbol "$\circ$" indicates that we discard one (any) column of the corresponding matrix, since we added a column representing the normalization condition. For $i \geq 2$, $\boldsymbol{\pi}^{(i)}$ can be obtained numerically from Eq.(3.1), but many useful performance metrics such as expected system utilization, throughput, or queue length can be expressed explicitly in closed-form using $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\mathbf{R}$ only (e.g., the average queue length is simply given by $\boldsymbol{\pi}^{(1)} \cdot (\mathbf{I}-\mathbf{R})^{-2} \cdot \mathbf{1}^T$) [64].

In the case of QBD processes, Eq.(3.2) simply reduces to the matrix quadratic equation

$$\mathbf{F} + \mathbf{R} \cdot \mathbf{L} + \mathbf{R}^2 \cdot \mathbf{B} = \mathbf{0},$$

while $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$ are obtained as the solution of the following system of linear equations [65]:

$$[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}] \cdot \begin{bmatrix} \mathbf{e} & (\mathbf{L}^{(0)})^\circ & \widehat{\mathbf{F}}^{(1)} \\ (\mathbf{I}-\mathbf{R})^{-1} \cdot \mathbf{e} & (\widehat{\mathbf{B}})^\circ & \mathbf{L} + \mathbf{R} \cdot \mathbf{B} \end{bmatrix} = [1, \mathbf{0}]. \qquad (3.4)$$

3.1.1 Additional measures of interest

Once the stationary probability vector $\boldsymbol{\pi}$ is known, we can obtain various performance measures of interest such as the average queue length which we showed in the previous subsection. There are additional measures of interest that one can compute knowing $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\mathbf{R}$. In particular, the tail distribution of the number of jobs in the system can be expressed as

$$\mathsf{P}[Ql > x] = \sum_{k=x+1}^{\infty} \boldsymbol{\pi}^{(k)}\mathbf{e} = \boldsymbol{\pi}^{(1)}\mathbf{R}^x(\mathbf{I}-\mathbf{R})^{-1}\mathbf{e}, \quad x \geq 0, \qquad (3.5)$$

with the corresponding expectation given by

$$\mathsf{E}[Ql] = \boldsymbol{\pi}^{(1)}\sum_{k=0}^{\infty}\mathbf{R}^k\mathbf{e}+\boldsymbol{\pi}^{(1)}\sum_{k=0}^{\infty}k\mathbf{R}^k\mathbf{e} = \boldsymbol{\pi}^{(1)}(\mathbf{I}-\mathbf{R})^{-1}\mathbf{e}+\boldsymbol{\pi}^{(1)}\mathbf{R}(\mathbf{I}-\mathbf{R})^{-2}\mathbf{e}. \quad (3.6)$$

The expected waiting time of a job in the system can then be calculated using Little's law [51] and Eq.(3.6), which yields

$$\mathsf{E}[W] = \lambda^{-1}\left(\boldsymbol{\pi}^{(1)}(\mathbf{I}-\mathbf{R})^{-1}\mathbf{e} + \boldsymbol{\pi}^{(1)}\mathbf{R}(\mathbf{I}-\mathbf{R})^{-2}\mathbf{e}\right). \quad (3.7)$$

Let η denote the spectral radius of the matrix $\mathbf{R}$, which is often called the *caudal characteristic* [68]. In addition to providing the stability condition for a QBD, η is indicative of the tail behavior of the stationary queue length distribution. Let $\mathbf{u}$ and $\mathbf{v}$ be the left and right eigenvectors corresponding to η normalized by $\mathbf{ue} = 1$ and $\mathbf{uv} = 1$. Under the above assumptions, it is known that [94]

$$\mathbf{R}^x = \eta^x\mathbf{v}\cdot\mathbf{u} + o(\eta^x), \quad \text{as } x \to \infty,$$

which together with Eq.(3.4) yields

$$\boldsymbol{\pi}^{(x)}\mathbf{e} = \boldsymbol{\pi}^{(1)}\mathbf{v}\,\eta^{x-1} + o(\eta^{x-1}), \quad \text{as } x \to \infty. \quad (3.8)$$

It then follows that

$$\mathsf{P}[Ql > x] = \frac{\boldsymbol{\pi}^{(1)}\mathbf{v}}{1-\eta}\eta^x + o(\eta^x), \quad \text{as } x \to \infty. \quad (3.9)$$

and thus

$$\lim_{x\to\infty} \frac{\mathsf{P}[Ql > x]}{\eta^x} = \frac{\boldsymbol{\pi}^{(1)}\mathbf{v}}{1-\eta}. \tag{3.10}$$

or equivalently

$$\mathsf{P}[Ql > x] \sim \frac{\boldsymbol{\pi}^{(1)}\mathbf{v}}{1-\eta}\eta^x, \quad \text{as } x \to \infty. \tag{3.11}$$

The caudal characteristic can be obtained without having to first solve for the matrix $\mathbf{R}$. We define the matrix $\mathbf{A}^*(s) = \mathbf{A}_0 + s\mathbf{A}_1 + s^2\mathbf{A}_2$, for $0 < s \leq 1$. Since the generator matrix $\mathbf{A}$ is irreducible, this matrix $\mathbf{A}^*(s)$ is irreducible with nonnegative off-diagonal elements. Let $\chi(s)$ denote the spectral radius of the matrix $\mathbf{A}^*(s)$. Then, under the above assumptions, η is the unique solution in $(0,1)$ of the equation $\chi(s) = 0$. A more efficient method is developed in [9].

3.2 Why does a geometric relation hold for QBD processes?

There is a clear intuitive appeal to the fact that a geometric relation holds for QBD processes. In this subsection, we first focus on the reasons for the existence of this relationship via a simple example. Our first example is a QBD process that models an $M/Cox_2/1$ queue. The state transition diagram of the CTMC that models this queue is depicted in Figure 3.1. The state space $\mathcal{S}$ of this CTMC is divided into subsets $\mathcal{S}^{(0)} = \{(0,0)\}$ and $\mathcal{S}^{(i)} = \{(i,1),(i,2)\}$ for $i \geq 1$, implying that the stationary probability vector is also divided in the respective subvectors $\boldsymbol{\pi}^{(0)} = [\boldsymbol{\pi}(0,0)]$

and $\pi^{(i)} = [\pi(i,1), \pi(i,2)]$, for $i \geq 1$. The block-partitioned infinitesimal generator

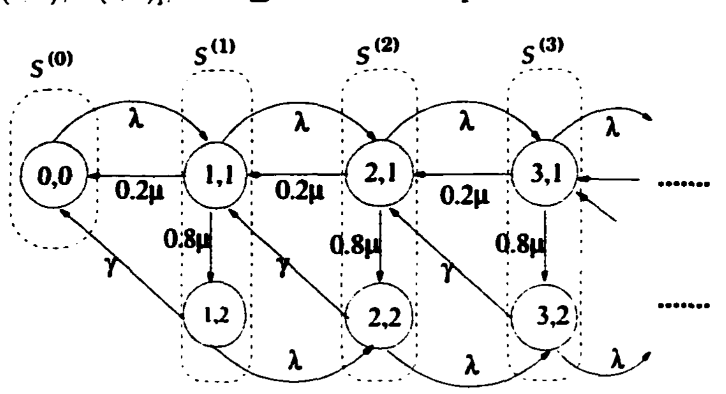

Figure 3.1: The CTMC modeling an $M/Cox_2/1$ queue.

$\mathbf{Q}_{QBD}$ is a infinite block tridiagonal matrix as defined in Eq.(2.18) and its component matrices are:

$$\widehat{\mathbf{L}} = [-\lambda], \qquad \widehat{\mathbf{F}} = [\lambda \quad 0], \qquad \widehat{\mathbf{B}} = \begin{bmatrix} 0.2\mu \\ \gamma \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} 0.2\mu & 0 \\ \gamma & 0 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} -(\lambda+\mu) & 0.8\mu \\ 0 & -(\lambda+\gamma) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}. \tag{3.12}$$

To illustrate the existence of the geometric relationship among the various stationary probability vectors $\pi^{(i)}$, we use the concept of stochastic complementation discussed in Subsection 2.8.2. We partition the state space into two subsets: $\mathcal{A} = \cup_{j=0}^{j=i}\mathcal{S}^{(j)}$ and $\overline{\mathcal{A}} = \cup_{j=i+1}^{\infty}\mathcal{S}^{(j)}$, i.e., a set with finite number of states and a set with an infinite number of states, respectively. The stochastic complement of the states in $\mathcal{A}$ is a new Markov chain that "skips over" all states in $\overline{\mathcal{A}}$. This Markov chain includes states in $\mathcal{A}$ only, but all transitions out of $\mathcal{S}^{(i)}$ (i.e., the boundary set) to $\mathcal{S}^{(i+1)}$ (i.e., the first set in $\overline{\mathcal{A}}$) need to be "folded back" to $\mathcal{A}$ (see Figure 3.2). This folding introduces a new direct transition with rate x that ensures that the stochastic

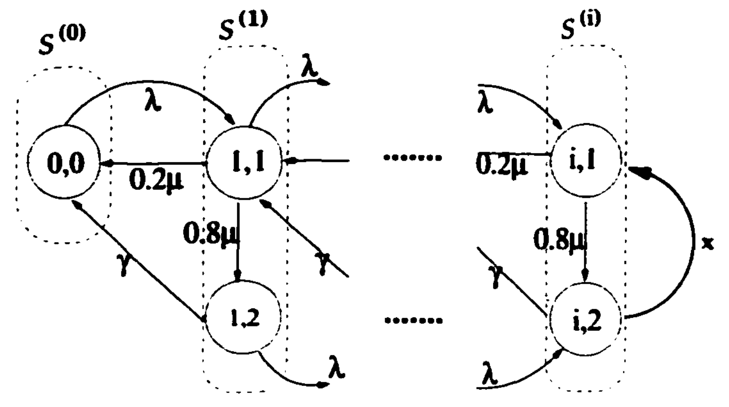

Figure 3.2: The CTMC of the stochastic complement of $\mathcal{A} = \cup_{j=0}^{j=i} S^{(j)}$ of the CTMC modeling an $M/Cox_2/1$ queue.

complement of the states in $\mathcal{A}$ is a stand-alone process. Because of the structure of this particular process, i.e., $\mathcal{A}$ is entered from $\overline{\mathcal{A}}$ *only* through state $(i, 1)$, x is simply equal to λ (see Lemma on single entry state in Subsection 2.8.2). Furthermore, because of the repetitive structure of the original chain, this rate does not depend on i (which essentially defines the size of set $\mathcal{A}$).

The steady state probability vector $\overline{\pi} = [\overline{\pi}^{(0)}, \cdots, \overline{\pi}^{(i)}]$ of the stochastic complement of the states in $\mathcal{A}$ relates to the steady state probability $\pi_{\mathcal{A}}$ of the original process with: $\overline{\pi} = \pi_{\mathcal{A}} / \pi_{\mathcal{A}} \cdot \mathbf{1}^T$. This implies that if a relation exists between $\overline{\pi}^{(i-1)}$ and $\overline{\pi}^{(i)}$, then the same relation holds for $\pi^{(i-1)}$ and $\pi^{(i)}$.

The flow balance equations for states $(i, 1)$ and $(i, 2)$ in the stochastic complement of $\mathcal{A}$, are:

$$(0.2\mu + 0.8\mu)\overline{\pi}^{(i)}[1] = \lambda\overline{\pi}^{(i-1)}[1] + \lambda\overline{\pi}^{(i)}[2],$$

$$(\gamma + \lambda)\overline{\pi}^{(i)}[2] = \lambda\overline{\pi}^{(i-1)}[2] + 0.8\mu\overline{\pi}^{(i)}[1],$$

which can further be expressed as:

$$(-\mu\pi^{(i)}[1] + \lambda\pi^{(i)}[2]) = -\lambda\pi^{(i-1)}[1],$$

$$(0.8\mu\pi^{(i)}[1] - (\gamma + \lambda)\pi^{(i)}[2]) = -\lambda\pi^{(i-1)}[2].$$

This last set of equations leads us to the following matrix equation

$$[\pi^{(i)}[1], \pi^{(i)}[2]] \begin{bmatrix} -\mu & 0.8\mu \\ \lambda & -(\gamma+\lambda) \end{bmatrix} = -[\pi^{(i-1)}[1], \pi^{(i-1)}[2]] \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix},$$

which implies that the relation between $\pi^{(i-1)}$ and $\pi^{(i)}$ can be expressed as

$$\pi^{(i)} = \pi^{(i-1)}\mathbf{R}, \tag{3.13}$$

where matrix $\mathbf{R}$ is defined as

$$\mathbf{R} = -\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \cdot \begin{bmatrix} -\mu & 0.8\mu \\ \lambda & -(\gamma+\lambda) \end{bmatrix}^{-1} = -\mathbf{F}\left(\mathbf{L} + \mathbf{F}\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}\right)^{-1}. \tag{3.14}$$

Applying Eq.(3.13) recursively, one can obtain the result of Eq.(3.1). Observe that in this particular case an explicit computation of $\mathbf{R}$ is possible (i.e., there is no need to compute $\mathbf{R}$ [77] via an iterative numerical procedure as in the general case). Explicit computation of $\mathbf{R}$ in this example is possible because backward transitions from $\mathcal{S}^{(i)}$ to $\mathcal{S}^{(i-1)}$ are directed toward a single state only. In Subsection 3.8, we give details on the cases when matrix $\mathbf{R}$ can be explicitly computed.

3.3 Generalization: geometric solution for the GI/M/1 processes

We generalize the finding in the previous example by considering a GI/M/1-queue with infinitesimal generator $\mathbf{Q}_{GI/M/1}$ similarly to the proof given in [45]. To evaluate the relation between $\boldsymbol{\pi}^{(i-1)}$ and $\boldsymbol{\pi}^{(i)}$ for $i > 1$, we construct the stochastic complement of the states in $\mathcal{A} = \cup_{j=0}^{i} \mathcal{S}^{(j)}$ ($\overline{\mathcal{A}} = \mathcal{S} - \mathcal{A}$). The stochastic complement of states in $\mathcal{A}$ has an infinitesimal generator defined by the following relation

$$\overline{\mathbf{Q}} = \mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot (-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1} \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}],$$

where

$$\mathbf{Q}[\mathcal{A},\mathcal{A}] = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \cdots & \mathbf{0} & \mathbf{0} \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{\mathbf{B}}^{(i-1)} & \mathbf{B}^{(i-2)} & \cdots & \mathbf{L} & \mathbf{F} \\ \widehat{\mathbf{B}}^{(i)} & \mathbf{B}^{(i-1)} & \cdots & \mathbf{B}^{(1)} & \mathbf{L} \end{bmatrix}, \quad \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \end{bmatrix},$$

$$\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}] = \begin{bmatrix} \widehat{\mathbf{B}}^{(i+1)} & \mathbf{B}^{(i)} & \cdots & \mathbf{B}^{(1)} \\ \widehat{\mathbf{B}}^{(i+2)} & \mathbf{B}^{(i+1)} & \cdots & \mathbf{B}^{(2)} \\ \widehat{\mathbf{B}}^{(i+3)} & \mathbf{B}^{(i+2)} & \cdots & \mathbf{B}^{(3)} \\ \widehat{\mathbf{B}}^{(i+4)} & \mathbf{B}^{(i+3)} & \cdots & \mathbf{B}^{(4)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}] = \begin{bmatrix} \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \cdots \\ \mathbf{B}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3.15}$$

Observe that $\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}]$ is the same matrix for any $i > 1$. We define its inverse to be as follows

$$(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1} = \begin{bmatrix} \mathbf{A}_{0.0} & \mathbf{A}_{0.1} & \mathbf{A}_{0.2} & \mathbf{A}_{0.3} & \cdots \\ \mathbf{A}_{1.0} & \mathbf{A}_{1.1} & \mathbf{A}_{1.2} & \mathbf{A}_{1.3} & \cdots \\ \mathbf{A}_{2.0} & \mathbf{A}_{2.1} & \mathbf{A}_{2.2} & \mathbf{A}_{2.3} & \cdots \\ \mathbf{A}_{3.0} & \mathbf{A}_{3.1} & \mathbf{A}_{3.2} & \mathbf{A}_{3.3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \qquad (3.16)$$

From the special structure of $\mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}]$ we conclude that the second term in the summation that defines $\overline{\mathbf{Q}}$ is a matrix with all block entries equal to zero except the very last block row, whose block entries $\mathbf{X}_j$ are of the form:

$$\mathbf{X}_j = \mathbf{F}\cdot\sum_{k=0}^{\infty}\mathbf{A}_{0.k}\widehat{\mathbf{B}}^{(j+1+k)} \qquad j = i$$

and

$$\mathbf{X}_j = \mathbf{F}\cdot\sum_{k=0}^{\infty}\mathbf{A}_{0.k}\mathbf{B}^{(j+1+k)}, \qquad 0 \le j < i.$$

Note that $\mathbf{X}_0 = \mathbf{F}\cdot\sum_{k=0}^{\infty}\mathbf{A}_{0.k}\mathbf{B}^{(1+k)}$ which means that $\mathbf{X}_0$ does not depend on the value of $i > 1$. The infinitesimal generator $\overline{\mathbf{Q}}$ of the stochastic complement of states in $\mathcal{A}$ is determined as

$$\overline{\mathbf{Q}} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \cdots & \mathbf{0} & \mathbf{0} \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{\mathbf{B}}^{(i-1)} & \mathbf{B}^{(i-2)} & \cdots & \mathbf{L} & \mathbf{F} \\ \widehat{\mathbf{B}}^{(i)}+\mathbf{X}_i & \mathbf{B}^{(i-1)}+\mathbf{X}_{i-1} & \cdots & \mathbf{B}^{(1)}+\mathbf{X}_1 & \mathbf{L}+\mathbf{X}_0 \end{bmatrix}. \qquad (3.17)$$

Let $\overline{\pi}$ be the stationary probability vector of the CTMC with infinitesimal generator $\overline{\mathbf{Q}}$ and $\pi_{\mathcal{A}}$ the steady-state probability vector of the CTMC of states in $\mathcal{A}$ in the original process, i.e., the process with infinitesimal generator $\mathbf{Q}_{GI/M/1}$. There is a linear relation between $\overline{\pi}$ and $\pi_{\mathcal{A}}$ given in the following equation:

$$\overline{\pi} = \frac{\pi_{\mathcal{A}}}{\pi_{\mathcal{A}} \cdot \mathbf{1}^T}. \tag{3.18}$$

Since $\overline{\pi}\overline{\mathbf{Q}} = \mathbf{0}$, we obtain the following relation

$$\overline{\pi}^{(i)} \cdot (\mathbf{L} + \mathbf{X}_0) = -\overline{\pi}^{(i-1)} \cdot \mathbf{F}$$

implying:

$$\pi^{(i)} \cdot (\mathbf{L} + \mathbf{X}_0) = -\pi^{(i-1)} \cdot \mathbf{F}.$$

The above equation holds for any $i > 1$, because their matrix coefficients do not depend on i. By applying it recursively over all vectors $\pi^{(i)}$ for $i > 1$, we obtain the following geometric relation

$$\pi^{(i)} = \pi^{(1)} \cdot \mathbf{R}^{i-1} \quad \forall\, i \geq 1.$$

Matrix $\mathbf{R}$, the geometric coefficient, has an important probabilistic interpretation: the entry (k, l) of $\mathbf{R}$ is the expected time spent in the state l of $\mathcal{S}^{(i)}$, before the first visit into $\mathcal{S}^{(i-1)}$, expressed in time unit Δ^i, given the starting state is k in $\mathcal{S}^{(i-1)}$. Δ^i is the mean sojourn time in the state k of $\mathcal{S}^{(i-1)}$ for $i \geq 2$ [67, pages 30-35].

3.4 Why M/G/1 processes are more difficult

For M/G/1-type processes there is no geometric relation among the various probability vectors $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$ as in the case of QBD and GI/M/1-type processes. In this section, we first demonstrate via a simple example why such a geometric relation does not exist and then we generalize and derive Ramaswami's recursive formula, i.e., the classic methodology for the solution of M/G/1 chains.

3.5 Example: a $BMAP_1/Cox_2/1$ queue

Figure 3.3 illustrates a Markov chain that models a $BMAP_1/Cox_2/1$ queue. This chain is very similar with the one depicted in Figure 3.1, the only difference is that the new chain models bulk arrivals of unlimited size. The infinitesimal generator $\mathbf{Q}_{M/G/1}$ of the process is block partitioned according to the partitioning of the state space $\mathcal{S}$ of this CTMC into subsets $\mathcal{S}^{(0)} = \{(0,0)\}$ and $\mathcal{S}^{(i)} = \{(i,1),(i,2)\}$ for $i \geq 1$. The definition of the component matrices of $\mathbf{Q}_{M/G/1}$ is as follows:

$$\widehat{\mathbf{L}} = [-2\lambda], \qquad \widehat{\mathbf{B}} = \begin{bmatrix} 0.2\mu \\ \gamma \end{bmatrix}, \qquad \widehat{\mathbf{F}}^{(i)} = [0.5^{i-1}\lambda \quad 0] \;\; i \geq 1,$$

$$\mathbf{B} = \begin{bmatrix} 0.2\mu & 0 \\ \gamma & 0 \end{bmatrix}, \; \mathbf{L} = \begin{bmatrix} -(2\lambda+\mu) & 0.8\mu \\ 0 & -(2\lambda+\gamma) \end{bmatrix}, \; \mathbf{F}^{(i)} = \begin{bmatrix} 0.5^{i-1}\lambda & 0 \\ 0 & 0.5^{i-1}\lambda \end{bmatrix} \; i \geq 1.$$

In the following, we derive the relation between $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$ and the rest of vectors in $\boldsymbol{\pi}$ using stochastic complementation, i.e., similarly to the approach described in Section 3.1. First we partition the state space $\mathcal{S}$ into two partitions $\mathcal{A} = \cup_{j=0}^{j=i}\mathcal{S}^{(j)}$

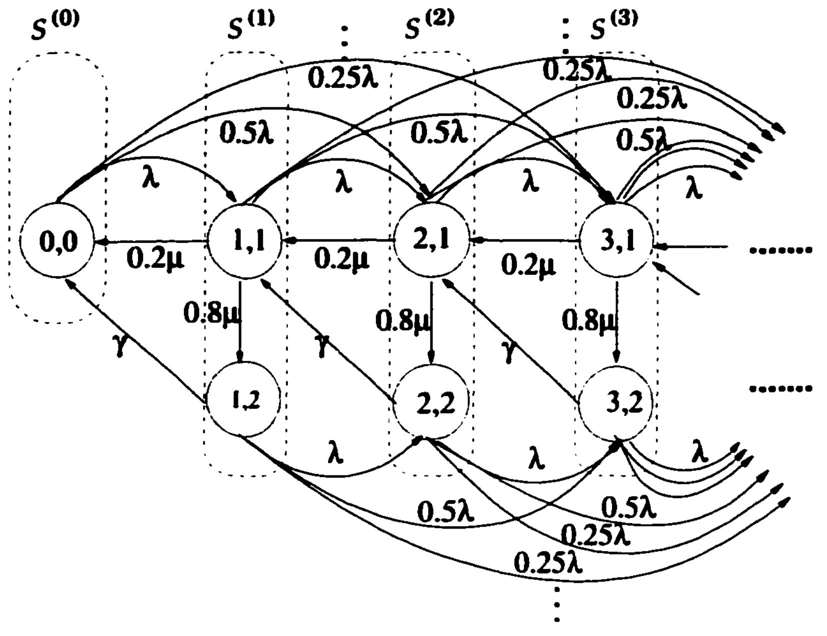

Figure 3.3: The CTMC that models a $BMAP_1/Cox_2/1$ queue.

and $\overline{\mathcal{A}} = \cup_{j=i+1}^{\infty} S^{(j)}$ and then we construct the stochastic complement of states in $\mathcal{A}$. The Markov chain of the stochastic complement of states in $\mathcal{A}$, (see Figure 3.4), illustrates how transitions from states $(j, 1)$ and $(j, 2)$ for $j \leq i$ and state $(0, 0)$ to states $(l, 1)$ and $(l, 2)$ for $l > i$ are folded back to state $(i, 1)$, which is the single state to enter $\mathcal{A}$ from states in $\overline{\mathcal{A}}$. These "back-folded" transitions are marked by $x_{k,h}$ for $k \leq i$ and $h = 1, 2$ and represent the "correction" needed to make the stochastic complement of states in $\mathcal{A}$, a stand-alone process. Because of the single entry state in $\mathcal{A}$ the stochastic complement of states in $\mathcal{A}$ for this example can be explicitly derived (see Lemma on single entry state in Subsection 2.8.2) and the definition of rates $x_{k,h}$ is as follows:

$$x_{k,h} = 2 \cdot 0.5^{i-k}\lambda = 0.5^{i-k-1}\lambda, \quad i \geq 1, \quad k \leq i, \quad h = 1, 2.$$

The flow balance equations for states $(i, 1)$ and $(i, 2)$ for the stochastic complement of states in $\mathcal{A}$ are:

$$\begin{aligned}(0.2\mu + 0.8\mu)\overline{\pi}^{(i)}[1] &= 2\lambda\overline{\pi}^{(i)}[2] \\ &+ 2 \cdot 0.5^{i-1}\lambda\overline{\pi}^{(0)}[1] \\ &+ 2 \cdot 0.5^{i-2}\lambda\overline{\pi}^{(1)}\ [1] + 0.5^{i-2}\lambda\overline{\pi}^{(1)}\ [2] + \ldots \\ &+ 2 \cdot 0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[1] + 0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[2]\end{aligned}$$

and

$$(2\lambda + \gamma)\overline{\pi}^{(i)}[2] = 0.8\mu\overline{\pi}^{(i)}[1] + 0.5^{i-2}\lambda\overline{\pi}^{(1)}[2] + \ldots + 0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[2].$$

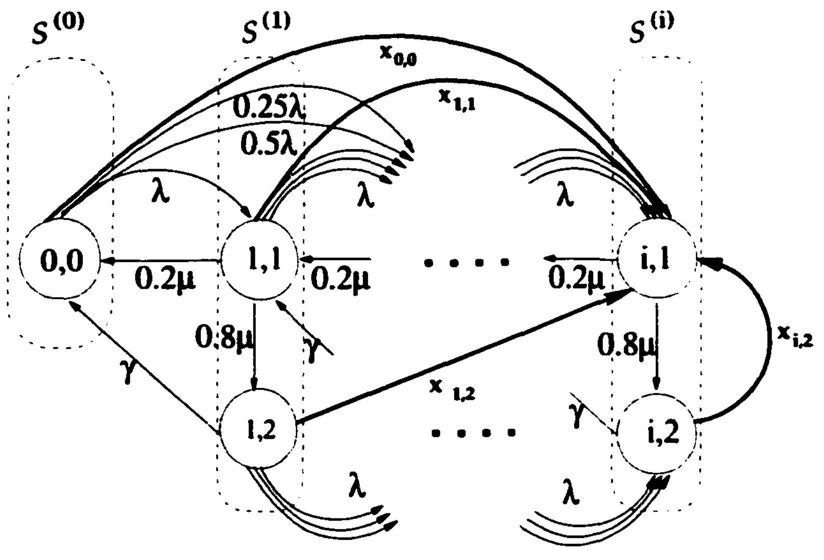

Figure 3.4: Stochastic complement of $BMAP_1/Cox_2/1$-type queue at level i.

In the above equalities, we group the elements of $\overline{\pi}^{(i)}$ on the left and the rest of the terms on the right in order to express their relation in terms of the block matrices that describe the infinitesimal generator $\overline{\mathbf{Q}}$ of the stochastic complement of states in

$\mathcal{A}$. By rearranging the terms, the above equations can be re-written as:

$$\begin{aligned}-\mu\overline{\pi}^{(i)}[1]+2\lambda\overline{\pi}^{(i)}[2] &= -(2\cdot 0.5^{i-1}\lambda\overline{\pi}^{(0)}[1]\\ &+ 2\cdot 0.5^{i-2}\lambda\overline{\pi}^{(1)}\;[1]+0.5^{i-2}\lambda\overline{\pi}^{(1)}\;[2]+...\\ &+ 2\cdot 0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[1]+0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[2])\end{aligned}$$

and

$$0.8\mu\overline{\pi}^{(i)}[1]-(2\lambda+\gamma)\overline{\pi}^{(i)}[2]=-(0.5^{i-2}\lambda\overline{\pi}^{(1)}[2]+...+0.5^{i-i}\lambda\overline{\pi}^{(i-1)}[2]).$$

We can now re-write the above equations in the following matrix equation form:

$$\begin{aligned}[\overline{\pi}^{(i)}[1],\ \overline{\pi}^{(i)}[2]]\cdot\begin{bmatrix}-\mu & 0.8\mu\\ 2\lambda & -(2\lambda+\gamma)\end{bmatrix} &= -(\overline{\pi}^{(0)}[1]\left[\ 2\cdot 0.5^{i-1}\lambda \quad 0\ \right]\\ &+ [\overline{\pi}^{(1)}\;[1],\ \overline{\pi}^{(1)}\;[2]]\begin{bmatrix}2\cdot 0.5^{i-2}\lambda & 0\\ 0.5^{i-2}\lambda & 0.5^{i-2}\lambda\end{bmatrix}+...\\ &+ [\overline{\pi}^{(i-1)}[1],\ \overline{\pi}^{(i-1)}[2]]\begin{bmatrix}2\cdot 0.5^{i-i}\lambda & 0\\ 0.5^{i-i}\lambda & 0.5^{i-i}\lambda\end{bmatrix}\Bigg).\end{aligned}$$

By substituting $[\overline{\pi}^{(i)}[1],\ \overline{\pi}^{(i)}[2]]$ with $\overline{\pi}^{(i)}$ and expressing the coefficient matrices in the above equation in terms of the component matrices of the infinitesimal generator $\overline{\mathbf{Q}}$ of the stochastic complement of states in $\mathcal{A}$, we obtain[3]:

$$\overline{\pi}^{(i)}\cdot(\mathbf{L}+\sum_{j=1}^{\infty}\mathbf{F}^{(j)}\mathbf{G})=-(\overline{\pi}^{(0)}\sum_{j=i}^{\infty}\widehat{\mathbf{F}}^{(j)}\mathbf{G}+\overline{\pi}^{(1)}\sum_{j=i-1}^{\infty}\mathbf{F}^{(j)}\mathbf{G}+...+\overline{\pi}^{(i-1)}\sum_{j=1}^{\infty}\mathbf{F}^{(j)}\mathbf{G}),$$

where $\mathbf{G}$ is a matrix with the following structure:

$$\mathbf{G}=\begin{bmatrix}1 & 0\\ 1 & 0\end{bmatrix}.$$

[3]Recall that $\overline{\pi}$ is the stationary probability vector of the stochastic complement of states in $\mathcal{A}$ and $\pi[\mathcal{A}]$ is the stationary probability vector of states in $\mathcal{A}$ in the original M/G/1 process. They relate to each other based on the equation $\overline{\pi}=\pi[\mathcal{A}]/(\pi[\mathcal{A}]\mathbf{1}^T)$, which implies that any relation that holds among subvectors $\overline{\pi}^{(j)}$ for $j\le i$ would hold for subvectors $\pi^{(j)}$ for $j\le i$ as well

Note that at this point, we have introduced a new matrix, $\mathbf{G}$, that has an important probabilistic interpretation. In this specific example, the matrix $\mathbf{G}$ can be explicitly derived [77]. This is a direct outcome of the fact that all states in set $\mathcal{S}^{(i)}$ for $i \geq 1$ return to the same *single* state in set $\mathcal{S}^{(i-1)}$. Equivalently, the matrix $\mathbf{B}$ of the infinitesimal generator $\mathbf{Q}_{M/G/1}$ has only a single column different from zero.

3.6 Generalization: derivation of Ramaswami's recursive formula

In this section, we investigate the relation between $\boldsymbol{\pi}^{(i)}$ for $i > 1$ and $\boldsymbol{\pi}^{(j)}$ for $0 \leq j < i$ for the general case in the same spirit as [93]. We construct the stochastic complementation of the states in $\mathcal{A} = \cup_{j=0}^{i} \mathcal{S}^{(j)}$ ($\overline{\mathcal{A}} = \mathcal{S} - \mathcal{A}$). We obtain

$$\mathbf{Q}[\mathcal{A},\mathcal{A}] = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} & \cdots & \widehat{\mathbf{F}}^{(i-1)} & \widehat{\mathbf{F}}^{(i)} \\ \widehat{\mathbf{B}} & \mathbf{L} & \cdots & \mathbf{F}^{(i-2)} & \mathbf{F}^{(i-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L} & \mathbf{F} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B} & \mathbf{L} \end{bmatrix}, \quad \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] = \begin{bmatrix} \widehat{\mathbf{F}}^{(i+1)} & \widehat{\mathbf{F}}^{(i+2)} & \widehat{\mathbf{F}}^{(i+3)} & \cdots \\ \mathbf{F}^{(i)} & \mathbf{F}^{(i+1)} & \mathbf{F}^{(i+2)} & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \mathbf{F}^{(4)} & \cdots \\ \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \end{bmatrix},$$

$$\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}] = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}] = \begin{bmatrix} \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The stochastic complement for states in $\mathcal{A}$ has an infinitesimal generator defined as follows

$$\overline{\mathbf{Q}} = \mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot (-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1} \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}].$$

Observe that $\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}]$ is the same matrix for any $i \geq 1$. We define its inverse to be as follows

$$(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} & \mathbf{A}_{0,2} & \mathbf{A}_{0,3} & \cdots \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \cdots \\ \mathbf{A}_{2,0} & \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & \cdots \\ \mathbf{A}_{3,0} & \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{3.19}$$

From the special structure of $\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]$, we conclude that the second term of the above summation is a matrix with all block entries equal to zero except the very last block column, whose block entries $\mathbf{X}_j$ are of the form:

$$\mathbf{X}_i = \sum_{k=0}^{\infty} \widehat{\mathbf{F}}^{(i+1+k)} \cdot \mathbf{A}_{k,0} \cdot \mathbf{B}$$

and

$$\mathbf{X}_j = \sum_{k=0}^{\infty} \mathbf{F}^{(j+1+k)} \cdot \mathbf{A}_{k,0} \cdot \mathbf{B}, \quad 0 \leq j < i.$$

The infinitesimal generator $\overline{\mathbf{Q}}$ of the stochastic complement of states in $\mathcal{A}$ is determined as

$$\overline{\mathbf{Q}} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} & \cdots & \widehat{\mathbf{F}}^{(i-1)} & \widehat{\mathbf{F}}^{(i)} + \mathbf{X}_i \\ \widehat{\mathbf{B}} & \mathbf{L} & \cdots & \mathbf{F}^{(i-2)} & \mathbf{F}^{(i-1)} + \mathbf{X}_{i-1} \\ \vdots & \vdots & & \vdots \quad \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L} & \mathbf{F}^{(1)} + \mathbf{X}_1 \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B} & \mathbf{L} + \mathbf{X}_0 \end{bmatrix}. \tag{3.20}$$

We define $\overline{\boldsymbol{\pi}}$ to be the steady-state probability vector of the CTMC with infinitesimal generator $\overline{\mathbf{Q}}$ and $\boldsymbol{\pi}_{\mathcal{A}}$ the steady-state probability vector of the CTMC with infinitesimal generator $\mathbf{Q}_{M/G/1}$ corresponding to the states in $\mathcal{A}$. There is a linear relation between $\overline{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}_{\mathcal{A}}$:

$$\overline{\boldsymbol{\pi}} = \frac{\boldsymbol{\pi}_{\mathcal{A}}}{\boldsymbol{\pi}_{\mathcal{A}} \cdot \mathbf{1}^T}. \tag{3.21}$$

From the relation $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$, it follows that

$$\boldsymbol{\pi}^{(i)} \cdot (\mathbf{L} + \mathbf{X}_0) = -(\boldsymbol{\pi}^{(0)} \cdot (\widehat{\mathbf{F}}^{(i)} + \mathbf{X}_i) + \sum_{j=1}^{i-1} \boldsymbol{\pi}^{(j)} \cdot (\mathbf{F}^{(i-j)} + \mathbf{X}_{i-j})) \quad \forall i \geq 1$$

and

$$\boldsymbol{\pi}^{(i)} \cdot (\mathbf{L} + \mathbf{X}_0) = -(\boldsymbol{\pi}^{(0)} \cdot (\widehat{\mathbf{F}}^{(i)} + \mathbf{X}_i) + \sum_{j=1}^{i-1} \boldsymbol{\pi}^{(j)} \cdot (\mathbf{F}^{(i-j)} + \mathbf{X}_{i-j})) \quad \forall i \geq 1. \quad (3.22)$$

The above equation shows that there in no geometric relation between vectors $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$, however it provides a recursive relation for the computation of the steady-state probability vector for M/G/1 Markov chains. In the following, we further work on simplifying the expression of matrices $\mathbf{X}_j$ for $0 \leq j \leq i$.

From the definition of the stochastic complementation (see Subsection 2.8.2), we know that an entry $[r, c]$ in $(-\mathbf{Q}[\overline{\mathcal{A}}, \overline{\mathcal{A}}]^{-1} \cdot \mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}])$[4] represents the probability that starting from state $r \in \overline{\mathcal{A}}$ the process enters $\mathcal{A}$ through state c. Since $\mathcal{A}$ is entered from $\overline{\mathcal{A}}$ only through states in $\mathcal{S}^{(i)}$, we can use the probabilistic interpretation of matrix $\mathbf{G}$ to figure out the entries in $(-\mathbf{Q}[\overline{\mathcal{A}}, \overline{\mathcal{A}}]^{-1}) \cdot \mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}]$. An entry $[r, c]$ in $\mathbf{G}^j$ for $j > 0$ represents the probability that starting from state $r \in \mathcal{S}^{(i+j)}$ for $i > 0$ the process enters set $\mathcal{S}^{(i)}$ through state c. It is straightforward now to define

$$(-\mathbf{Q}[\overline{\mathcal{A}}, \overline{\mathcal{A}}]^{-1}) \cdot \mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}] = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G}^1 \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G}^2 \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G}^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \quad (3.23)$$

[4]Only the entries of the last block column of $(-\mathbf{Q}[\overline{\mathcal{A}}, \overline{\mathcal{A}}]^{-1}) \cdot \mathbf{Q}[\overline{\mathcal{A}}, \mathcal{A}]$ are different from zero.

The above result simplifies the expression of $\mathbf{X}_j$ as follows

$$\mathbf{X}_i = \sum_{k=1}^{\infty} \widehat{\mathbf{F}}^{(i+k)} \cdot \mathbf{G}^k \text{ and } \mathbf{X}_j = \sum_{k=1}^{\infty} \mathbf{F}^{(j+k)} \cdot \mathbf{G}^k, \quad 0 \leq j < i. \tag{3.24}$$

This is in essence Ramaswami's recursive formula. We will return to this in the following section after we elaborate on matrix **G**, its implications, and its probabilistic interpretation.

3.7 General solution of M/G/1-type processes

For the solution of M/G/1-type processes, several algorithms exist [12, 60, 69]. These algorithms first compute matrix **G** as the solution of the following equation:

$$\mathbf{B} + \mathbf{L}\mathbf{G} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \mathbf{G}^{i+1} = \mathbf{0}. \tag{3.25}$$

The matrix **G** has an important probabilistic interpretation: an entry (r, c) in **G** expresses the conditional probability of the process first entering $\mathcal{S}^{(i-1)}$ through state c, given that it starts from state r of $\mathcal{S}^{(i)}$ [69, page 81][5]. Figure 3.5 illustrates the relation of entries in **G** for different paths of the process. From the probabilistic interpretation of **G** the following structural properties hold [69]

- if the M/G/1 process with infinitesimal generator $\mathbf{Q}_{M/G/1}$ is recurrent then **G** is row-stochastic.

[5]The probabilistic interpretation of **G** is the same for both DTMCs and CTMCs.

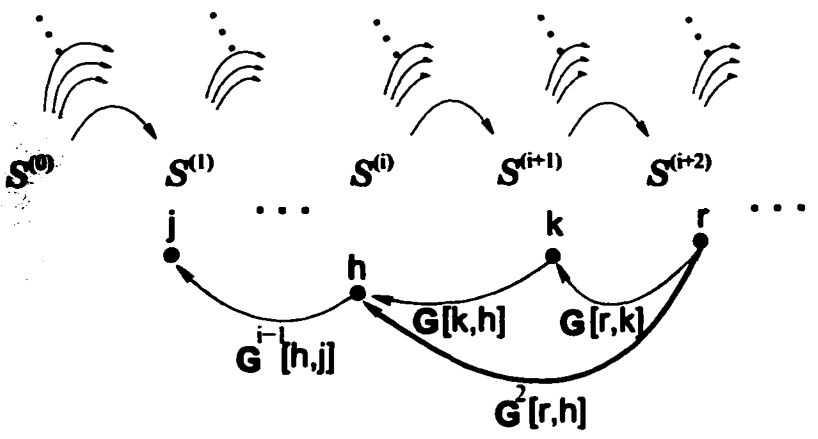

Figure 3.5: Probabilistic interpretation of **G**.

- to any zero column in matrix **B** of the infinitesimal generator $\mathbf{Q}_{M/G/1}$, there is a corresponding zero column in matrix **G**.

The **G** matrix is obtained by solving iteratively Eq.(3.25). However, recent advances show that the computation of **G** is more efficient when displacement structures are used based on the representation of M/G/1-type processes by means of QBD processes [60, 12, 11, 47]. The most efficient algorithm for the computation of **G** is the cyclic reduction algorithm [12].

3.7.1 Ramaswami's formula

From Eqs.(3.22) and (3.24) and the aid of matrix **G**, we derive Ramaswami's recursive formula [75], which is numerically stable because it entails only additions and multiplications[6]. Ramaswami's formula defines the following recursive relation among

[6]Subtractions on these type of formulas present the possibility of numerical instability [69, 75].

stationary probability vectors $\boldsymbol{\pi}^{(i)}$ for $i \geq 0$:

$$\boldsymbol{\pi}^{(i)} = -\left(\boldsymbol{\pi}^{(0)}\widehat{\mathbf{S}}^{(i)} + \sum_{k=1}^{i-1}\boldsymbol{\pi}^{(k)}\mathbf{S}^{(i-k)}\right)\mathbf{S}^{(0)^{-1}} \quad \forall i \geq 1, \tag{3.26}$$

where, letting $\mathbf{F}^{(0)} \equiv \mathbf{L}$, matrices $\widehat{\mathbf{S}}^{(i)}$ and $\mathbf{S}^{(i)}$ are defined as follows:

$$\widehat{\mathbf{S}}^{(i)} = \sum_{l=i}^{\infty}\widehat{\mathbf{F}}^{(l)}\mathbf{G}^{l-i}, \; i \geq 1, \quad \mathbf{S}^{(i)} = \sum_{l=i}^{\infty}\mathbf{F}^{(l)}\mathbf{G}^{l-i}, \; i \geq 0. \tag{3.27}$$

Observe that the above auxiliary sums represent the last column in the infinitesimal generator $\overline{\mathbf{Q}}$ defined in Eq.(3.20). We can express them in terms of matrices $\mathbf{X}_i$ defined in Eq.(3.24) as follows:

$$\widehat{\mathbf{S}}^{(i)} = \widehat{\mathbf{F}}^{(i)} + \mathbf{X}_i, \; i \geq 1 \quad \mathbf{S}^{(i)} = \mathbf{F}^{(i)} + \mathbf{X}_i, \; i \geq 0.$$

Given the above definition of $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$ and the normalization condition, a unique vector $\boldsymbol{\pi}^{(0)}$ can be obtained by solving the following system of m linear equations, i.e., the cardinality of set $\mathcal{S}^{(0)}$:

$$\boldsymbol{\pi}^{(0)}\left[\left(\widehat{\mathbf{L}}^{(0)} - \widehat{\mathbf{S}}^{(1)}\mathbf{S}^{(0)^{-1}}\widehat{\mathbf{B}}\right)^{\circ} \mid \mathbf{1}^T - \sum_{i=1}^{\infty}\widehat{\mathbf{S}}^{(i)}\left(\sum_{j=0}^{\infty}\mathbf{S}^{(j)}\right)^{-1}\mathbf{1}^T\right] = [\mathbf{0} \mid 1], \tag{3.28}$$

where the symbol "$\circ$" indicates that we discard one (any) column of the corresponding matrix, since we added a column representing the normalization condition. Once $\boldsymbol{\pi}^{(0)}$ is known, we can then iteratively compute $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$, stopping when the accumulated probability mass is close to one. After this point, measures of interest can be computed. Since the relation between $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$ is not straightforward, computation of measures of interest require generation of the entire stationary probability vector.

3.7.2 Explicit computation of G

A special case of M/G/1-type processes occurs when $\mathbf{B}$ is a product of two vectors, i.e., $\mathbf{B} = \boldsymbol{\alpha} \cdot \boldsymbol{\beta}$. Assuming, without loss of generality, that $\boldsymbol{\beta}$ is normalized, then $\mathbf{G} = \mathbf{1}^T \cdot \boldsymbol{\beta}$, i.e., it is derived explicitly [77, 78].

For this special case, $\mathbf{G} = \mathbf{G}^n$, for $n \geq 1$. This special structure of matrix $\mathbf{G}$ simplifies the form of matrices $\widehat{\mathbf{S}}^{(i)}$ for $i \geq 1$, and $\mathbf{S}^{(i)}$ for $i \geq 0$ defined in Eq.(3.27):

$$\begin{array}{l} \widehat{\mathbf{S}}^{(i)} = \widehat{\mathbf{F}}^{(i)} + (\sum_{j=i+1}^{\infty} \widehat{\mathbf{F}}^{(j)}) \cdot \mathbf{G}, \quad i \geq 1 \\ \mathbf{S}^{(i)} = \widehat{\mathbf{F}}^{(i)} + (\sum_{j=i+1}^{\infty} \mathbf{F}^{(j)}) \cdot \mathbf{G}, \quad i \geq 0, \quad \mathbf{F}^{(0)} \equiv \mathbf{L}. \end{array} \tag{3.29}$$

In this special case, $\mathbf{G}$ does not need to be either computed or fully stored, which is a considerable gain since in an M/G/1-type process computation of G is expensive and G needs to be stored throughout the solution procedure.

3.7.3 Fast FFT Ramaswami's formula

Meini in [58] gives an improved version of Ramaswami's formula. Once $\boldsymbol{\pi}^{(0)}$ is known using Eq.(3.28), the stationary probability vector is computed using matrix-generating functions associated with block triangular Toeplitz matrices[7]. These matrix-

[7]A Toeplitz matrix has equal elements in each of its diagonals allowing the use of computationally efficient methods.

generating functions are computed efficiently by using fast Fourier transforms (FFT).

The algorithm of the Fast FFT Ramaswami's formula is based on the fact that in practice it is not possible to store an infinite number of matrices to express the M/G/1-type process. Assuming that only p matrices can be stored then the infinitesimal generator $\mathbf{Q}_{M/G/1}$ has the following structure

$$\mathbf{Q}_{M/G/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} & \widehat{\mathbf{F}}^{(2)} & \widehat{\mathbf{F}}^{(3)} & \widehat{\mathbf{F}}^{(4)} & \cdots & \widehat{\mathbf{F}}^{(p)} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots & \mathbf{F}^{(p-1)} & \mathbf{F}^{(p)} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots & \mathbf{F}^{(p-2)} & \mathbf{F}^{(p-1)} & \mathbf{F}^{(p)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots & \mathbf{F}^{(p-3)} & \mathbf{F}^{(p-2)} & \mathbf{F}^{(p-1)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \qquad (3.30)$$

Because there are only p matrices of type $\widehat{\mathbf{F}}^{(i)}$ and $\mathbf{F}^{(i)}$, there are only p sums of type $\widehat{\mathbf{S}}^{(i)}$ and $\mathbf{S}^{(i)}$ to be computed. Therefore, the computation of $\boldsymbol{\pi}^{(i)}$ for $i >$ 0 using Ramaswami's formula, i.e., Eq.(3.26), depends only on p vectors $\boldsymbol{\pi}^{(j)}$ for $max(0, i-p) \leq j < i$. Define

$$\tilde{\boldsymbol{\pi}}^{(1)} = [\boldsymbol{\pi}^{(1)}, \dots, \boldsymbol{\pi}^{(p)}] \quad \text{and} \quad \tilde{\boldsymbol{\pi}}^{(i)} = [\boldsymbol{\pi}^{(p(i-1)+1)}, \dots, \boldsymbol{\pi}^{(pi)}] \quad \text{for} \quad i \geq 2. \qquad (3.31)$$

The above definition simplifies the formalization of Ramaswami's formula since $\tilde{\boldsymbol{\pi}}^{(i)}$ depends only on the values of $\tilde{\boldsymbol{\pi}}^{(i-1)}$ for $i > 1$. If we apply Ramaswami's formula for vectors $\pi^{(1)}$ to $\pi^{(p)}$, we obtain the following equations

$$\begin{array}{l}
\pi^{(1)}=-\pi^{(0)}\widehat{\mathbf{S}}^{(1)}(\mathbf{S}^{(0)})^{-1} \\
\pi^{(2)}=-(\pi^{(0)}\widehat{\mathbf{S}}^{(2)}+\pi^{(1)}\mathbf{S}^{(1)})(\mathbf{S}^{(0)})^{-1} \\
\pi^{(3)}=-(\pi^{(0)}\widehat{\mathbf{S}}^{(3)}+\pi^{(1)}\mathbf{S}^{(2)}+\pi^{(2)}\mathbf{S}^{(1)})(\mathbf{S}^{(0)})^{-1} \\
\vdots \\
\pi^{(p)}=-(\pi^{(0)}\widehat{\mathbf{S}}^{(p)}+\pi^{(1)}\mathbf{S}^{(p-1)}+\ldots+\pi^{(p-1)}\mathbf{S}^{(1)})(\mathbf{S}^{(0)})^{-1}
\end{array}. \tag{3.32}$$

We rewrite the above equations in the following form:

$$\begin{array}{ll}
\pi^{(1)}\mathbf{S}^{(0)} & =-\pi^{(0)}\widehat{\mathbf{S}}^{(1)} \\
\pi^{(2)}\mathbf{S}^{(0)}+\pi^{(1)}\mathbf{S}^{(1)} & =-\pi^{(0)}\widehat{\mathbf{S}}^{(2)} \\
\pi^{(3)}\mathbf{S}^{(0)}+\pi^{(2)}\mathbf{S}^{(1)}+\pi^{(1)}\mathbf{S}^{(2)} & =-\pi^{(0)}\widehat{\mathbf{S}}^{(3)} \\
\vdots & \\
\pi^{(p)}\mathbf{S}^{(0)}+\pi^{(p-1)}\mathbf{S}^{(1)}+\ldots+\pi^{(1)}\mathbf{S}^{(p-1)}=-\pi^{(0)}\widehat{\mathbf{S}}^{(p)} &
\end{array}. \tag{3.33}$$

Define

$$\mathbf{Y}=\begin{bmatrix}
\mathbf{S}^{(0)} & \mathbf{S}^{(1)} & \mathbf{S}^{(2)} & \cdots & \mathbf{S}^{(p-1)} \\
\mathbf{0} & \mathbf{S}^{(0)} & \mathbf{S}^{(1)} & \cdots & \mathbf{S}^{(p-2)} \\
\mathbf{0} & \mathbf{0} & \mathbf{S}^{(0)} & \cdots & \mathbf{S}^{(p-3)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}^{(0)}
\end{bmatrix} \quad \text{and} \quad \mathbf{b}=\left[\widehat{\mathbf{S}}^{(1)}, \widehat{\mathbf{S}}^{(2)}, \widehat{\mathbf{S}}^{(3)}, \cdots, \widehat{\mathbf{S}}^{(p)}\right]. \tag{3.34}$$

The set of equations in Eq.(3.33) can be written in a compact way by using the definitions in Eq.(3.31) and Eq.(3.34).

$$\tilde{\pi}^{(1)}=-\pi^{(0)}\cdot\mathbf{b}\cdot\mathbf{Y}^{-1}. \tag{3.35}$$

We apply Ramswami's formula for all vectors $\pi^{(j)}$, $p(i-1)+1\le j\le pi$ in $\tilde{\pi}^{(i)}$ for $i>1$.

$$\begin{array}{lll}
\pi^{(p(i-1)+1)}=-(\pi^{(p(i-2)+1)}\mathbf{S}^{(p)}+\ldots+ & \pi^{(p(i-1))}\mathbf{S}^{(1)}) & (\mathbf{S}^{(0)})^{-1} \\
\pi^{(p(i-1)+2)}=-(\pi^{(p(i-2)+2)}\mathbf{S}^{(p)}+\ldots+ & \pi^{(p(i-1)+1)}\mathbf{S}^{(1)}) & (\mathbf{S}^{(0)})^{-1} \\
\pi^{(p(i-1)+3)}=-(\pi^{(p(i-2)+3)}\mathbf{S}^{(p)}+\ldots+ & \pi^{(p(i-1)+2)}\mathbf{S}^{(1)}) & (\mathbf{S}^{(0)})^{-1} \\
\vdots & & \\
\pi^{(p(i-1)+p)}=-(\pi^{(p(i-2)+p)}\mathbf{S}^{(p)}+\ldots+\pi^{(p(i-1)+p-1)}\mathbf{S}^{(1)}) & & (\mathbf{S}^{(0)})^{-1}
\end{array}.$$

These equations can be rewritten in the following form

$$\begin{array}{ll}
\pi^{(p(i-1)+1)}\mathbf{S}^{(0)} & =-(\pi^{(p(i-2)+1)}\mathbf{S}^{(p)}+...+\pi^{(p(i-1))}\mathbf{S}^{(1)}) \\
\pi^{(p(i-1)+2)}\mathbf{S}^{(0)}+\pi^{(p(i-1)+1)}\mathbf{S}^{(1)} & =-(\pi^{(p(i-2)+2)}\mathbf{S}^{(p)}+...+\pi^{(p(i-1))}\mathbf{S}^{(2)}) \\
\pi^{(p(i-1)+3)}\mathbf{S}^{(0)}+...+\pi^{(p(i-1)+1)}\mathbf{S}^{(2)} & =-(\pi^{(p(i-2)+3)}\mathbf{S}^{(p)}+...+\pi^{(p(i-1))}\mathbf{S}^{(3)}) \\
\vdots & \\
\pi^{(p(i-1)+p)}\mathbf{S}^{(0)}+...+\pi^{(p(i-1)+1)}\mathbf{S}^{(p-1)}=-\pi^{(p(i-1))}\mathbf{S}^{(p)} &
\end{array}.$$

The above set of equations can be written in a matrix form as

$$\tilde{\pi}^{(i)}=-\tilde{\pi}^{(i-1)}\cdot\mathbf{Z}\mathbf{Y}^{-1} \qquad i\geq 2, \tag{3.36}$$

where matrix $\mathbf{Y}$ is defined in Eq.(3.34) and the definition of matrix $\mathbf{Z}$ is given by the following

$$\mathbf{Z}=\begin{bmatrix} \mathbf{S}^{(p)} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^{(p-1)} & \mathbf{S}^{(p)} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{S}^{(2)} & \mathbf{S}^{(3)} & \cdots & \mathbf{S}^{(p)} & \mathbf{0} \\ \mathbf{S}^{(1)} & \mathbf{S}^{(2)} & \cdots & \mathbf{S}^{(p-1)} & \mathbf{S}^{(p)} \end{bmatrix}. \tag{3.37}$$

The Fast Ramaswami's Formula consists of the set of equations defined in Eq.(3.35) and Eq.(3.36). The effectiveness of the representation of the Ramaswami's formula in the form of Eq.(3.35) and Eq.(3.36) comes from the special structure of matrices $\mathbf{Y}$ and $\mathbf{Z}$. The matrix $\mathbf{Y}$ is an upper block triangular Toeplitz matrix and the matrix $\mathbf{Z}$ is a lower block triangular Toeplitz matrix. Using fast Fourier transforms one can compute efficiently the inverse of a Toeplitz matrix or the multiplication of a vector with a Toeplitz matrix [58]. Although the use of Fourier transforms for matrix operations may result in numerical instabilities, in numerous test cases the above algorithm has not experienced instability [58, 59].

3.8 Explicit computation of R for QBDs

QBD processes are defined as the intersection of M/G/1 and GI/M/1-type processes. Hence, both matrix **G** (characteristic for M/G/1) and matrix **R** (characteristic for GI/M/1) can be defined for a QBD process as solutions of the following quadratic equations [47]:

$$\mathbf{B} + \mathbf{L}\mathbf{G} + \mathbf{F}\mathbf{G}^2 = \mathbf{0}, \qquad \mathbf{F} + \mathbf{R}\mathbf{L} + \mathbf{R}^2\mathbf{B} = \mathbf{0}.$$

If matrix-geometric is used to solve a QBD process then the relation between $\boldsymbol{\pi}^{(i)}$ and $\boldsymbol{\pi}^{(i-1)}$ for $i > 1$ is expressed in terms of **R**

$$\boldsymbol{\pi}^{(i)} = \boldsymbol{\pi}^{(i-1}\mathbf{R},$$

If matrix-analytic is the solution method then the relation between $\boldsymbol{\pi}^{(i)}$ and $\boldsymbol{\pi}^{(i-1)}$ is based on Ramaswami's recursive formula:

$$\boldsymbol{\pi}^{(i)} = -\boldsymbol{\pi}^{(i-1)}\mathbf{S}^{(1)}(\mathbf{S}^{(0)})^{-1},$$

where $\mathbf{S}^{(1)} = \mathbf{F}$ and $\mathbf{S}^{(0)} = (\mathbf{L} + \mathbf{F}\mathbf{G})$, i.e., the only auxiliary sums (see Subsection 3.7.1) used in the solution of M/G/1 processes that are defined for a QBD process. The above equations allow the derivation of the fundamental relation between **R** and **G** [47, pages 137-8],

$$\mathbf{R} = -\mathbf{F}(\mathbf{L} + \mathbf{F}\mathbf{G})^{-1}. \tag{3.38}$$

Obviously, for the case of QBD processes, knowing $\mathbf{G}$ (or $\mathbf{R}$) implies a direct computation of $\mathbf{R}$ (or $\mathbf{G}$). Computing $\mathbf{G}$ is usually easier than computing $\mathbf{R}$: $\mathbf{G}$'s computation is a prerequisite to the computation of $\mathbf{R}$ in the logarithmic reduction algorithm, the most efficient algorithm to compute $\mathbf{R}$ [47]. If $\mathbf{B}$ can be expressed as a product of two vectors

$$\mathbf{B} = \boldsymbol{\alpha} \cdot \boldsymbol{\beta},$$

where, without loss of generality $\boldsymbol{\beta}$ is assumed to be a normalized vector, then $\mathbf{G}$ and $\mathbf{R}$ can be explicitly obtained as

$$\mathbf{G} = \mathbf{1} \cdot \boldsymbol{\beta}, \quad \mathbf{R} = -\mathbf{F}(\mathbf{L} + \mathbf{F}\mathbf{1} \cdot \boldsymbol{\beta})^{-1}.$$

Representative examples, where the above condition holds, are the queues $M/Cox/1$, $M/Hr/1$, and $M/Er/1$, whose service process is Coxian, Hyperexponential, and Erlang distribution respectively.

The other case of explicit computation of $\mathbf{R}$ in QBD processes is when the matrix $\mathbf{F}$ of the infinitesimal generator matrix in Eq.(2.18) is given by

$$\mathbf{F} = \mathbf{w} \cdot \boldsymbol{\beta}, \tag{3.39}$$

where $\mathbf{w}$ is a column vector and $\boldsymbol{\beta}$ is a row vector, and without loss of generality it is assumed that $\boldsymbol{\beta} \cdot \mathbf{1}^T = 1$. In this case, the rate matrix $\mathbf{R}$ can be computed from

$$\mathbf{R} = -\mathbf{F}(\mathbf{L} + \eta\mathbf{B})^{-1} = \mathbf{w} \cdot \boldsymbol{\xi}, \tag{3.40}$$

where $\boldsymbol{\xi} = -\boldsymbol{\beta} \cdot (\mathbf{L} + \eta\mathbf{B})^{-1}$, and η is the spectral radius, or the maximal eigenvalue of $\mathbf{R}$. There are at least two algorithms to compute η, one is given by Neuts [67, pages 36-40], and the other which explores the special structure of $\mathbf{R}$ is given in [77].

3.9 Conditions for stability

We briefly review the conditions that enable us to assert that the CTMC described by the infinitesimal generator $\mathbf{Q}_{M/G/1}$ in Eq.(2.22) is stable, that is, admits a probability vector satisfying $\boldsymbol{\pi}\mathbf{Q}_{M/G/1} = \mathbf{0}$ and $\boldsymbol{\pi}\mathbf{1}^T = 1$.

First observe that the matrix $\tilde{\mathbf{Q}} = \mathbf{B} + \mathbf{L} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)}$ is an infinitesimal generator, since it has zero row sums and non-negative off-diagonal entries. The conditions for stability depend on the irreducibility of matrix $\tilde{\mathbf{Q}}$.

If $\tilde{\mathbf{Q}}$ is irreducible then there exists a unique positive vector $\tilde{\boldsymbol{\pi}}$ that satisfies the equations $\tilde{\boldsymbol{\pi}}\tilde{\mathbf{Q}} = \mathbf{0}$ and $\tilde{\boldsymbol{\pi}}\mathbf{1}^T = 1$. In this case, the stability condition for the M/G/1-type process with infinitesimal generator $\mathbf{Q}_{M/G/1}$ [69] is given by the following inequality

$$\tilde{\boldsymbol{\pi}}(\mathbf{L} + \sum_{j=1}^{\infty}(j+1)\mathbf{F}^{(j)})\mathbf{1}^T = \tilde{\boldsymbol{\pi}}(\mathbf{L} + \sum_{j=1}^{\infty}\mathbf{F}^{(j)})\mathbf{1}^T + \tilde{\boldsymbol{\pi}}\sum_{j=1}^{\infty} j\mathbf{F}^{(j)}\mathbf{1}^T < 0.$$

Since $\mathbf{B} + \mathbf{L} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)}$ is an infinitesimal generator, then $(\mathbf{B} + \mathbf{L} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)})\mathbf{1}^T = 0$. By substituting in the condition for stability the term $(\mathbf{L} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)})\mathbf{1}^T$ with its equal

$(-\mathbf{B}\mathbf{1}^T)$, the condition for stability can be re-written as:

$$\tilde{\boldsymbol{\pi}} \sum_{j=1}^{\infty} j\mathbf{F}^{(j)}\mathbf{1}^T < \tilde{\boldsymbol{\pi}}\mathbf{B}\mathbf{1}^T. \tag{3.41}$$

As in the scalar case, the equality in the above relation results in a null-recurrent CTMC.

In the example of the $BMAP_1/Cox_2/1$ queue depicted in Figure 3.3, the infinitesimal generator $\tilde{\mathbf{Q}}$ and its stationary probability vector are

$$\tilde{\mathbf{Q}} = \begin{bmatrix} -0.8\mu & 0.8\mu \\ \gamma & -\gamma \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{\pi}} = \left[\frac{\gamma}{\gamma+0.8\mu}, \ \frac{0.8\mu}{\gamma+0.8\mu}\right],$$

while

$$\sum_{j=1}^{\infty} j\mathbf{F}^{(j)} = \begin{bmatrix} 4\lambda & 0 \\ 0 & 4\lambda \end{bmatrix}.$$

The stability condition is expressed as

$$\left[\frac{\gamma}{\gamma+0.8\mu}, \ \frac{0.8\mu}{\gamma+0.8\mu}\right] \cdot \begin{bmatrix} 4\lambda & 0 \\ 0 & 4\lambda \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} < \left[\frac{\gamma}{\gamma+0.8\mu}, \ \frac{0.8\mu}{\gamma+0.8\mu}\right] \cdot \begin{bmatrix} 0.2\mu & 0 \\ \gamma & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

which can be written in the following compact form

$$4\lambda < \frac{\mu \cdot \gamma}{0.8\mu + \gamma}.$$

If $\tilde{\mathbf{Q}}$ is reducible, then the stability condition is different. By identifying the absorbing states in $\tilde{\mathbf{Q}}$, its state space can be rearranged as follows

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_k & \mathbf{0} \\ \mathbf{D}_1 & \mathbf{D}_2 & \cdots & \mathbf{D}_k & \mathbf{D}_0 \end{bmatrix}, \qquad (3.42)$$

where the blocks $\mathbf{C}_h$ for $1 \leq h \leq k$ are irreducible and infinitesimal generators. Since the matrices $\mathbf{B}$, $\mathbf{L}$ and $\mathbf{F}^{(i)}$ for $i \geq 1$ have non-negative off-diagonal elements, they can be restructured similarly and have block components $\mathbf{B}_{\mathbf{C}_h}$, $\mathbf{B}_{\mathbf{D}_l}$, $\mathbf{L}_{C_h}$, $\mathbf{L}_{D_l}$, and $\mathbf{F}^{(i)}_{C_h}$, $\mathbf{F}^{(i)}_{D_l}$ for $1 \leq h \leq k$, $0 \leq l \leq k$, and $i \geq 1$.

This implies that each of the sets $\mathcal{S}^{(i)}$ for $i \geq 1$ is partitioned into subsets that communicate only through the boundary portion of the process, i.e., states in $\mathcal{S}^{(0)}$. The stability condition in Eq.(3.41) should be satisfied by all the irreducible blocks identified in Eq.(3.42) in order for the M/G/1-type process to be stable as summarized below:

$$\tilde{\boldsymbol{\pi}}^{(h)} \sum_{j=1}^{\infty} j\mathbf{F}^{(j)}_{C_h}\mathbf{1}^T < \tilde{\boldsymbol{\pi}}^{(h)}\mathbf{B}_{\mathbf{C}_h}\mathbf{1}^T \qquad \forall 1 \leq h \leq k. \qquad (3.43)$$

3.10 Chapter summary

In this chapter, we derived the basic matrix analytic results for the solution of M/G/1-type, GI/M/1-type, and QBD processes. Via simple examples and from first principles, we illustrated why the solution of QBD and GI/M/1-type processes is simpler than the solution of M/G/1-type processes. We presented the classic solution techniques for such processes, the elegant matrix-geometric and the matrix-analytic methods. We gave an overview of the recent advances in matrix-analytic methodology, concentrating on the most efficient algorithms for computation of G and $\mathbf{R}$ and the FFT Ramamswami's formula. We direct the interested reader in the two books of Neuts [67, 69] for further details, as well as to the book of Latouche and Ramaswami [47].

Chapter 4

Data Fitting Algorithms

In this chapter, we present three new fitting techniques which approximate data sets with PH distributions and Markovian arrival processes, i.e., stochastic processes that are tractable. Our objective is to fit data sets that exhibit long-tailed behavior and long-range dependence into PH distributions and MAPs, respectively. In our approach, we strive for both accuracy and efficiency. We achieve these goals by developing divide-and-conquer and hierarchical fitting algorithms. The fitting algorithms leverage the use of the matrix-analytic methodology (see Chapter 3), for the performance analysis of Internet-related systems. We evaluate the accuracy of our fitting techniques not only statistically but also from the queueing systems perspective, because the intention is to use our fitted models as inputs in queueing models.

This chapter is organized as follows. In Section 4.1, we elaborate on the complexity of systems with highly variable service process, commonly encountered in Internet systems. In Section 4.2, we present a new technique for fitting long-tailed data sets into PH distributions by partitioning the data into smaller subsets of equal variability.

fitting each of the subsets into PH distributions, and combining results together. In Section 4.3, we propose a similar fitting technique with the exception that the partitioning of the data is based on the expected value of each subset. In Section 4.4, we devise a technique for fitting long-range dependent data sets into MAPs. We conclude with a summary of chapter's contributions.

4.1 Long-tailed behavior

There is abundance of evidence [3, 50] that the service process in Internet-related systems is heavy-tailed. This characteristic affects the complexity of such systems (e.g., Web servers, routers). As a motivation, we first present the effects of high variability on user perceived performance by analyzing the behavior of an elementary queueing system whose service process is long-tailed. We use an M/G/1-type queue, i.e., a single server queue with general service process, admitting Markovian arrivals. We run several experiments, that are distinguished from each other by the characteristics of the service process only: the mean of the service process, i.e., the first moment, is fixed, while the coefficient of variation, i.e., the second moment, varies within a certain range. The M/G/1 queue is solved using the matrix-geometric method outlined in Chapter 3.

Figure 4.1 illustrates the effects of the service process variability on the expected mean slowdown[1] as a function of the arrival rates. As the arrival rate in the system

[1]The mean slowdown metric is computed by dividing the average waiting time in the queue with

increases, the average slowdown for the M/G/1 queue with the highest variability in the service process increases much faster than the average slowdown of the queues with less variability in their service process. The same behavior is also observed in Figure 4.1(b), where we plot the average queue length as a function of the variability in the service process. We observe that for high arrival rates the effect of the highly variable service process on performance is more severe than for the case of low arrival rates. This simple example clearly shows that capturing the variability in the service process is essential for accurate performance analysis.

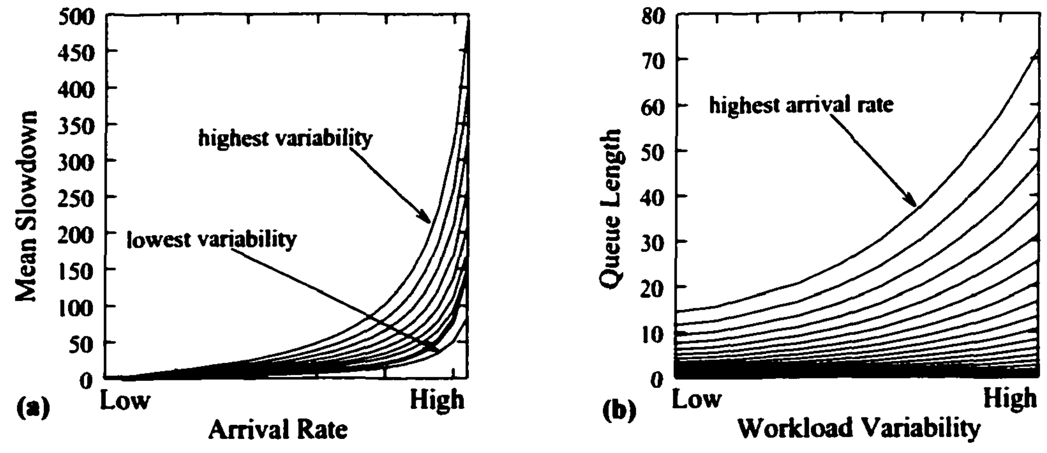

Figure 4.1: Effects in the performance of queuing system with highly variable service process.

Long-tailed behavior in a data set is commonly approximated by distributions such as Lognormal, Weibull, and Pareto. The analysis of queueing systems that admit these distributions as either arrival or service process is complex and, usually, intractable. PH distributions (along with their special case of Coxian and hyperexpo-

the average service time.

nential distributions) offer an attractive alternative that can capture variability via a tractable distribution. Their only "drawback" is that they require more parameters than Lognormal or Weibull distributions, making their estimation procedures more complex. To avoid the complicated fitting algorithms and yet benefit from their tractability, modelers often use Coxian or hyperexponential distributions with only two phases. In such cases, there are only few, i.e., 2 or 3, parameters to estimate, and the fitting is usually done by matching the first and the second moments only [43, pages 141-143]. Below we show that a simple PH distribution, such as a two-phase hyperexponential model, does not capture the complex behavior of the service process in Internet-related systems.

To fit an empirical distribution with a certain mean and CV into a 2-phase hyperexponential distribution, which is a mixture of two exponentials only, we need to calculate the following parameters: p $(0 \leq p \leq 1)$, λ_1, and λ_2, as the probability density function of a hyperexponential distribution is given by:

$$h_2(x) = p\lambda_1 e^{-\lambda_1 x} + (1-p)\lambda_2 e^{-\lambda_2 x}, x \geq 0, \quad \lambda_1, \lambda_2 > 0.$$

One can match the mean m_d and the coefficient of variation cv_d of the data set with the mean m_h and the coefficient of variation cv_h of the hyperexponential distribution, using the following formulas [43, pages 141-143]:

$$m_d = m_h = \frac{p}{\lambda_1} + \frac{1-p}{\lambda_2} \quad \text{and} \quad cv_d^2 = cv_h^2 = \frac{\frac{2p}{\lambda_1^2} + \frac{2(1-p)}{\lambda_2^2}}{m_d^2} - 1.$$

This is a classic case of a system of two equations and three unknowns. This system can be solved by guessing one of its unknowns (in this case the easiest one to guess is p since $0 \leq p \leq 1$). To illustrate that such a simple model does not capture the complex behavior of the queueing systems with heavy-tailed service process, we fit a data set into a two-phase hyperexponential using the above method. The data consists of the sizes of the requested files processed by a Web server (i.e., one entire day of server logs from the 1998 World Soccer Cup site, whose characteristics we evaluate in more detail later in this dissertation). Assuming that the requests arrive to the system according to a Poisson process and the service process is determined by the sizes of the requested files, we model the Web server as an $M/H_2/1$ queue. To assess the accuracy of the results obtained from the analytic solution of this queueing model (solved using the matrix-geometric method outlined in Chapter 3), we simulate the same single server queue using an identical arrival process but now the service process is driven by the original trace data. We present our findings in Figure 4.2. First, in Figure 4.2(a), we present the average queue length in the server as function of the arrival rate. We observe that for different guessed values of p the models compute exactly the same average queue length for each arrival rate and match the respective simulations result. Note that for performance metrics such as the average queue length, simple models are sufficient. We go a step further and analyze the queue length distribution. We present the body and the tail of the queue length distribution (measured for high server utilization) in Figures 4.2(b) and (c), respectively. None of the queue length distributions of the analytic models follows the shape of the queue length distribution obtained by the trace-driven simulation.

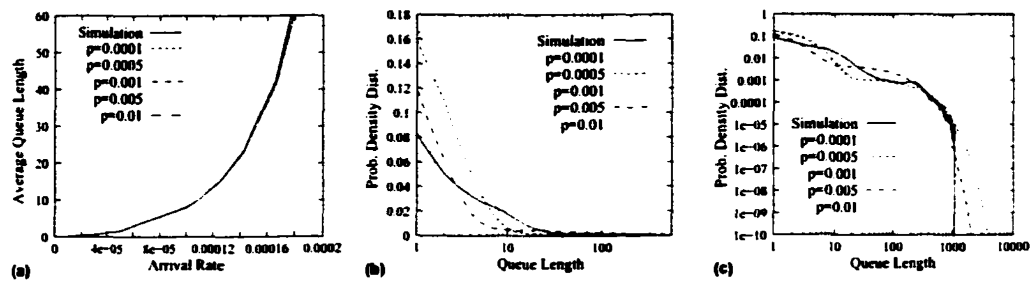

Figure 4.2: Average queue length, the body, and the tail of the queue length distribution (80% server utilization).

With the examples presented in this section, we emphasize the complexity in the behavior of systems that operate under a highly variable service process and the importance of capturing it correctly for accurate performance analysis results. In the following sections, we present fitting techniques that capture correctly the long-tailed behavior of the data sets and the respective queueing systems.

4.2 D&C EM

In this section, we present a new technique for fitting highly variable data sets into hyperexponential distributions. The hyperexponential distribution is characterized by the number of exponential phases *ph*, the mean λ_i, and probability p_i associated with each phase (as defined in Section 2.6). The proposed methodology applies the EM algorithm, described in Subsection 2.6.1.1, in a divide and conquer fashion over the initial data set. We call our approach D&C EM (Divide-and-Conquer-EM).

The high level idea of the proposed method (see Figure 4.3) is based on the observation that for data sets that exhibit long-tailed behavior, it may be beneficial to partition their entire range of values so as to ensure that each partition exhibits significantly reduced variability in comparison to the variability of the entire data set. The data set of each partition is then fitted into a hyperexponential distribution using the EM algorithm [72] and the final fit for the entire data set is generated by combining together the fitting results for all partitions.

The divide and conquer approach increases the accuracy of the EM algorithm because the portion of the data set belonging to the tail of its continuous data histogram (CDH) [48] could possibly fit in one or more partitions, reducing the possibility that the EM algorithm does not capture it correctly while searching for the global optimal solution. The approach is efficient because each partition has less data entries and less variability than the entire data set, facilitating an accurate fit in a few phases only.

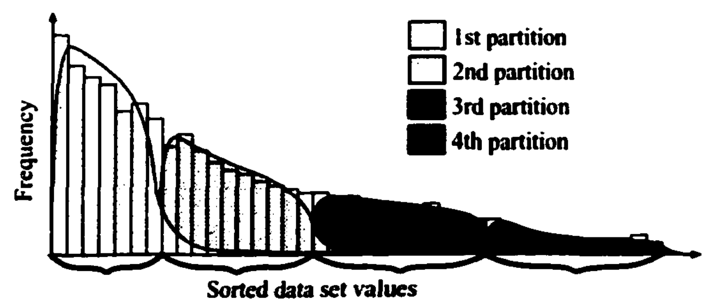

Figure 4.3: Splitting of the continuous data histogram (CDH).

We start by building the CDH with only one pass through the data. One additional pass through the CDH is required to define the "splits". Since we are interested to split the CDH such that each partition has reduced variability, we use the coefficient of variation (CV) to determine the partition boundaries. For each partition, we accumulate bins until the accumulated coefficient of variation for that partition, CV_{Acc}, is larger than a threshold, CV_{max}. The value of CV_{max} determines the number of partitions for a given data set. We select CV_{max} to be between 1.2 and 1.5, i.e., slightly higher than the CV of the exponential distribution, in order to fit each partition into a hyperexponential distribution with few phases only using EM. The EM algorithm requires as input only the number of phases, ph, and the actual data. In our experiments, we use $ph = 4$ since it captures well the properties of data sets with CVs as high as CV_{Max} and strikes a good balance between efficiency and accuracy.

We generate the final result by combining together the weight of each partition to the entire CDH with its respective fitted hyperexponential distribution. Since the total number of phases is obtained as the sum of phases over all partitions, the final number of phases from the fitting is not known until the fitting procedure is complete. From numerical experiments, we observe that data sets with high CV may contain as many as 6 partitions, while data sets with not-as-high CVs may contain as low as 3 partitions. Figure 4.4 summarizes the D&C EM algorithm.

4.2.1 D&C EM experimental results

In this section, we present results obtained by fitting five different data sets into hyperexponential distributions using the D&C EM algorithm. We first describe the

```
1. Build CDH from the data set.
2. while (there are still CDH bins to be considered)
     a include data of current bin into current partition $i$
     b update $CV_{Acc}$
     c if $(CV_{Acc} > CV_{max})$
             Use EM to fit partition $i$ into $ph$ phases
             Obtain $p^i_j$ and $\lambda^i_j$, $1 \le j \le ph$
             Compute weight $w_i$ for partition $i$
             $CV_{Acc} = 0$.
3. if $(CV_{last\ partition} < CV_{max})$
     a Merge last two partitions and perform step 2c
4. Generate final result
     for i from 1 to # of partitions
       for j from 1 to $ph$
         $p^i_j = p^i_j \cdot w_i$
```

Figure 4.4: D&C EM fitting algorithm.

characteristics of the selected data sets.

4.2.1.1 Traces

We have selected five highly variable data sets to test our approach. The first data set (indicated as "Trace 1") is a trace from the 1998 World Soccer Cup Web site[2]. It contains the sizes of the files requested by clients from this Web site in the course of an entire day. The other four traces are synthetically generated from analytic models that closely approximate Web server traffic [3]. Traces 2 and 3 are generated from Lognormal distributions with shape parameters 1.85 and 1.5, respectively, and the same scale parameter 7.0. Traces 4 and 5 are generated from Weibull distributions

[2]Available at http://ita.ee.lbl.gov/.

with shape parameters 0.25 and 0.35, respectively, and the same scale parameter 9.2. The statistical characteristics of these data sets are shown in Table 4.1. The number

Trace	Entries	Unique	Mean	CV
1	16045065	12122	4407.81	7.28
2	25000	25000	6358.23	5.87
3	25000	25000	3459.86	3.13
4	25000	22969	227.27	7.36
5	25000	24298	47.50	3.86

Table 4.1: Statistical characteristics of the data sets.

of entries and the number of unique entries for each data set are significant for the performance of the D&C EM since the running time of the EM algorithm depends on these parameters [72]. Observe that the real trace has less unique entries than the synthetically generated data sets.

4.2.1.2 D&C EM statistical analysis

The size of the data sets precludes using goodness-of-fit tests such as the Kolmogorov-Smirnov and χ^2 tests [48]. Therefore, we evaluate the accuracy of D&C EM by checking the matching of statistical properties such as the moments and the median, and by plotting PDFs, CDFs, and CCDFs (Complimentary Cumulative Distribution or Survival function).

Table 4.2 illustrates the means and the CVs of the original data sets, plus various hyperexponential fittings using the EM, FW, and D&C EM algorithms. In Table 4.2, "x ph" means that the EM or FW algorithms fit the entire data set into a hyperexponential with x phases. Observe that the D&C EM models match the mean of the traces with maximal error of 4%. The coefficient of variation is more difficult to match, since it is obtained using both the first and the second moments. Nevertheless, D&C EM models match it with a maximal error of 20% (Trace 2). The EM algorithm alone could not generate results for Traces 4 and 5 within reasonable amount of computation time (less than a week) in a Pentium III 800MHz processor with 1GB of memory. Since Traces 4 and 5 are synthetically generated from a Weibull distributions, we fit the same distribution functions into hyperexponential distributions using the FW algorithm and compare them with the D&C EM fits. The results of Table 4.2 show that D&C EM technique matches better the statistical properties of the data sets, when compared to the EM and FW algorithms.

D&C EM fits match better even the higher moments of the data sets. We present in Table 4.3 the relative errors of fitted third moments from the actual third moments of all five data sets (we omit the absolute values because they are too large and not easy to read).

Figure 4.5 plots the PDH, CDF, and CCDF for each data set and the D&C EM, EM, and FW models. The PDF plots for all five data sets are shown in the first column of graphs in Figure 4.5 (note the logscale of the x-axis). The PDF of Trace 1 is heavily jagged, characteristic of real trace data, which makes matching the PDF more challenging. D&C EM offers accurate fits for all traces. The fits for Traces 2 and 3, both D&C EM and EM ones, do not match well the body of the data

	Data	EM 8 ph	FW 15 ph	D&C EM
Trace 1				
Mean	4407.81	4402.35	n/a	4393.56
CV	7.28	3.44	n/a	7.86
Median	938.00	961.51	n/a	950.59
Trace 2				
Mean	6358.23	6196.27	n/a	6164.50
CV	5.87	4.13	n/a	5.13
Median	1082.91	1063.12	n/a	1061.25
Trace 3				
Mean	3459.86	3425.72	n/a	3391.06
CV	3.13	2.69	n/a	2.82
Median	1085.32	1084.13	n/a	1086.59
Trace 4				
Mean	227.27	n/a	221.34	220.21
CV	7.36	n/a	8.12	7.14
Median	2.20	n/a	2.06	2.27
Trace 5				
Mean	47.50	n/a	46.40	46.61
CV	3.86	n/a	3.95	3.87
Median	3.32	n/a	3.14	3.35

Table 4.2: Statistical evaluation of the fittings.

	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5
EM 8 ph	75%	91%	70%	n/a	n/a
FW 15 ph	n/a	n/a	n/a	106%	27%
D&C EM	60%	60%	66%	25%	11%

Table 4.3: Relative error of the third moment.

PDFs. This happens because Traces 2 and 3 do not have monotone PDFs, while the hyperexponential distribution has a completely monotone PDF [30].

The CDF plots for all traces (middle column of graphs in Figures 4.5) illustrate

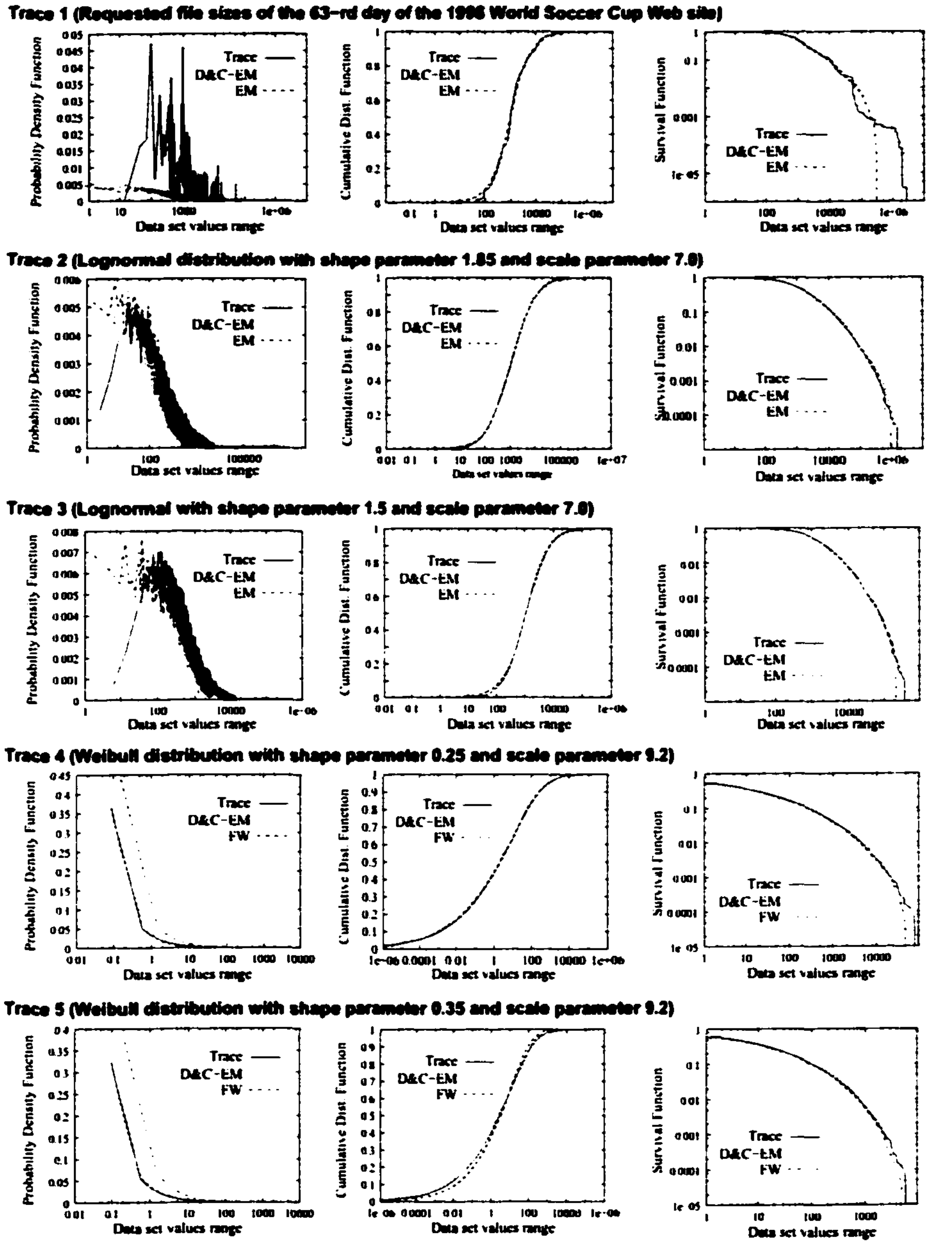

Figure 4.5: PDF, CDF, and CCDF of the real data and the fitted models.

that the D&C EM provides a good match for the body of the distribution for all traces. In order to investigate the accuracy of the fittings for the tail of the distribution, we present the CCDF plots in log-log scale (third column of graphs in Figure

4.5). Note that even for the tail of the distribution, which reflects the observed variability of the data sets, D&C EM generates models that closely match the data set characteristics.

4.2.2 D&C EM queueing system analysis

Because we want to provide a methodology that allows for performance analysis of Internet-related systems, we also examine the accuracy of the D&C EM from a queueing systems perspective. We consider an $M/H_r/1$ server queue with exponentially distributed interarrival times and service times drawn from a hyperexponential distribution. The hyperexponential model for the service process is generated from the test data sets using the D&C EM, EM, or FW algorithms. We opt for exponential interarrival times in order to concentrate on the effects of the service process *only* on queueing behavior.

To analyze the $M/H_r/1$ queue that resulted from our fittings, we used the matrix-geometric method presented in Section 3.1. The matrix-geometric solution provides the entire stationary probability distribution for the queueing system under study, the average queue length, and the queue length distribution. In our analysis, we focus on both the average queue length and the queue length distribution. The queue length distribution is an important metric because it can guide system design and at the same time is a strong indicator of model accuracy. Furthermore, because our focus are long-tailed data sets, we need to demonstrate that the proposed hyperexponential server captures well the tail of the queue length distribution.

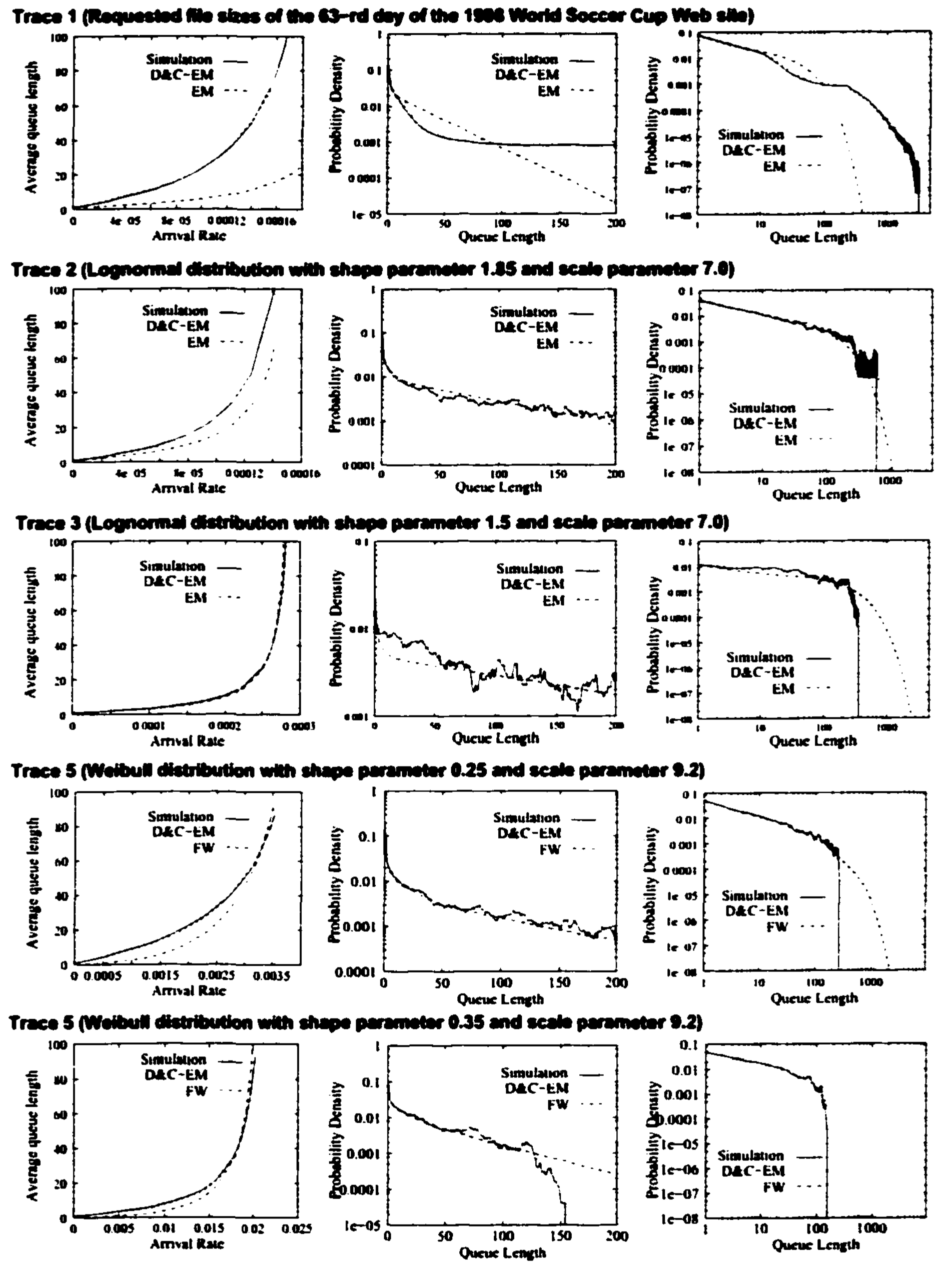

Figure 4.6: Queue length, body and tail of queue length distribution for 80% system utilization.

To examine the accuracy of the fitting algorithm, we compare the above performance measures, that are analytically derived using the $M/H_r/1$ queue, with those obtained from trace-driven simulations. The simulation model consists of a single server queue with the same arrival process as in the $M/H_r/1$ model and service times from the data sets of Table 4.1. Results are presented in Figure 4.6. For all data sets, we plot the average queue length as function of the arrival rate (first column of graphs in Figure 4.6), the body of the queue length distribution (middle column in Figure 4.6), and the tail of the queue length distribution (last column in Figure 4.6). We obtain the queue length tail distribution by plotting the queue length distribution in a log-log scale. The queue length distributions in Figure 4.6 correspond to 80% system utilization levels.

Observe that the models obtained from D&C EM fittings generate queueing system results that are close to the simulation results. The $M/H_r/1$ queueing system captures accurately the performance metrics of interest. Consistent with the discussion in [30], we note that for Traces 4 and 5, whose CDH is completely monotone, the hyperexponential distribution is a better approximation, than for Traces 1, 2, and 3, whose CDH is not completely monotone.

4.2.3 Computational efficiency of D&C EM

In this section, we report on the computational efficiency of D&C EM for fitting data sets into hyperexponential distributions. In Figure 4.7, we illustrate the computation time needed to obtain fittings for the data sets indicated in Table 4.1 using the D&C

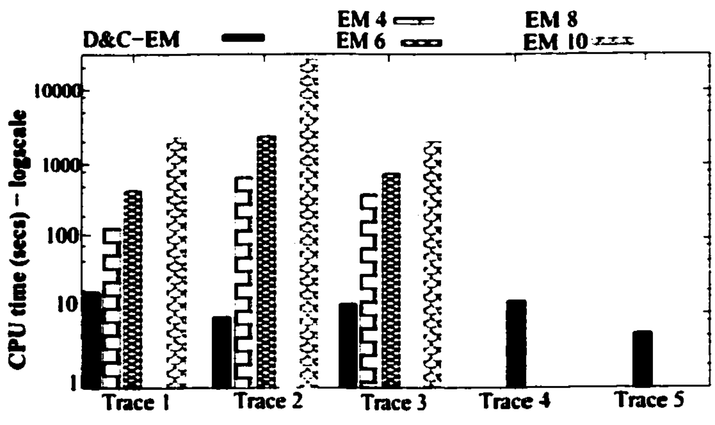

Figure 4.7: CPU time for running EM and D&C EM on all the traces.

EM and EM algorithms. The experiments were conducted in a Linux machine with Pentium III 800MHz processor and 1GB of memory. D&C EM is much faster than EM, even for fittings that consist of few phases only. The efficiency of D&C EM increases as the complexity of the data set increases, i.e., longer tails and more unique entries. Note that for Traces 4 and 5, we could not obtain results even for 4-phase models within a time frame of a week, while we obtained the D&C EM fits in a matter of seconds.

4.2.4 Refined D&C EM fitting algorithm

The hyperexponential distribution has a complete monotone PDF. Hence it provides better fits for data sets and distribution functions whose PDFs are complete monotone. Example of distribution functions with complete monotone PDFs are Pareto and Weibull distribution (with parameters as in Traces 4 and 5). Traces 1, 2, and 3 do not have complete monotone CDHs (equivalent to PDF of a distribution). For the data sets with non-monotone CDHs, the hyperexponential distribution is not the best choice among the special cases of PH distributions. A PH distribution constructed as a mixture of an Erlang and a hyperexponential distribution is a better fit for data sets with CDH shapes similar to Traces 1, 2, and 3, i.e., non-monotone with only one spike in the PDF [98] (shown also in Figure 4.8).

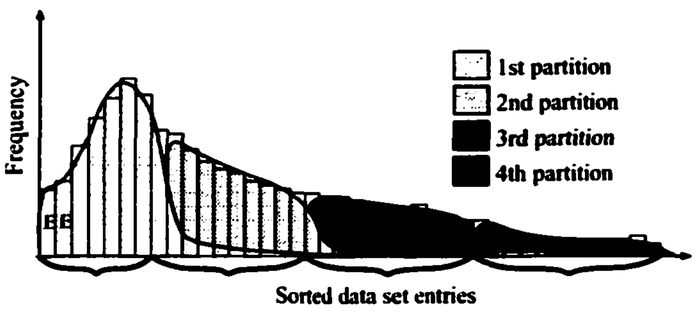

Figure 4.8: Splitting of the non-monotone continuous data histogram (CDH).

In order to fit the data set into a mixture of Erlang and hyperexponential distributions, we first determine the portion of the data set CDH that we need to fit into an Erlang. We start from the first bin of the CDH and accumulate bins, in the same fashion described in Section 4.2, until we take into consideration approximately 0.5% of the data set entries. The value 0.5% accounts for just a "small" portion in the beginning of the CDH[3]. We denote the index of the last bin of this accumulation with Max. In Figure 4.8, the first two bins marked with "E" illustrate how we accumulate bins for the Erlang fitting. In the example of Figure 4.8, Max is equal to 2. The accumulated data is fitted into a 2-phase Erlang. Since the Erlang distribution with

[3]The fitting algorithm is not sensitive to the amount of collected bins for the Erlang fitting, since we tried values in (0.1%, 1%) and obtain similar results.

2 phases has only one parameter, i.e., the mean λ_{Er} of each of the exponential phases, we use the moment matching method [43] rather than the EM algorithm to determine this parameter.

```
1. Build CDH from the data set.
2. Find bin index $Max$ such that:
     a entries upto bin $Max$ account for 0.5% of all entries,
     b bin frequencies increase.
3. if ($Max > 0$)
     a Fit the first $Max$ CDH bins into a 2-phase Erlang
     b Obtain $\lambda_{Er}$
4. while (there are still CDH bins to be considered)
     a include data of current bin into current partition $i$
     b update $CV_{Acc}$
     c if ($CV_{Acc} > CV_{max}$)
             Use EM to fit partition $i$ into $ph$ phases
             Obtain $p^i_j$ and $\lambda^i_j$, $1 \le j \le ph$
             Compute weight $w_i$ for partition $i$
             $CV_{Acc} = 0$.
5. if ($CV_{last\ partition} < CV_{max}$)
     a Merge last two partitions and perform step 4c
6. Generate the final hyperexponential fit
     for i from 1 to # of partitions
       for j from 1 to $ph$
         $p^i_j = p^i_j \cdot w_i$
7. if ($Max > 0$)
     a Obtain the final PH fit by merging
      the Erlang and the hyperexponential fits
```

Figure 4.9: Refined D&C EM fitting algorithm.

Once the parameter of the Erlang distribution is determined, we continue with the D&C EM algorithm as described in Figure 4.4 to fit the entire data set into a hyperexponential distribution. Upon completion of the fitting procedure, we merge the Erlang and hyperexponential fits by letting the Erlang fit proceed the hyperexponential one in the final PH distribution. We illustrate the structure of vector $\boldsymbol{\tau}$ and matrix $\mathbf{T}$ for the PH distribution in Eq.(4.1), assuming 3-phase hyperexponential and 2-phase Erlang fits. Recall that $\boldsymbol{\tau}$ and $\mathbf{T}$ are the parameters of a PH distribution as defined in Subsection 2.6.

$$\boldsymbol{\tau} = [1,\ 0,\ 0,\ 0,\ 0] \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} -\lambda_{Er} & \lambda_{Er} & 0 & 0 & 0 \\ 0 & -\lambda_{Er} & p_1\lambda_{Er} & p_2\lambda_{Er} & p_3\lambda_{Er} \\ 0 & 0 & -\lambda_3 & 0 & 0 \\ 0 & 0 & 0 & -\lambda_4 & 0 \\ 0 & 0 & 0 & 0 & -\lambda_5 \end{bmatrix}, \quad (4.1)$$

where p_i for $1 \leq i \leq 3$ are the probabilities associated with each phase of the hyperexponential distribution.

Figure 4.9 summarizes the refined D&C EM algorithm that allows for fitting data sets with not completely monotone CDHs into PH distributions. Observe that while we introduce three new steps, i.e., **Step 2**, **3**, and **7**, we adjust the refined D&C EM algorithm to accommodate both cases of monotone and non-monotone densities. **Step 2** determines if the CDH is non-monotone and an Erlang fit is required. We emphasize that the refined D&C EM algorithm provides improvements in fitting the body of the CDH and preserves the tail-matching accuracy and computational efficiency of the original D&C EM.

4.2.4.1 Results with the refined D&C EM

In this subsection, we present experimental results obtained by applying the refined D&C EM algorithm to fit Traces 1, 2, and 3, whose CDHs are not completely monotone. In Figure 4.10, we present the probability densities for Traces 1, 2, and 3, and the D&C EM fits obtained using both versions of D&C EM algorithm. The first row represent the hyperexponential fits and the second row the Erlang-hyperexponential fits. Observe that the refined D&C EM algorithm captures better the shape of the body of the CDH even for Trace 1 which is heavily jagged. We illustrate the moment

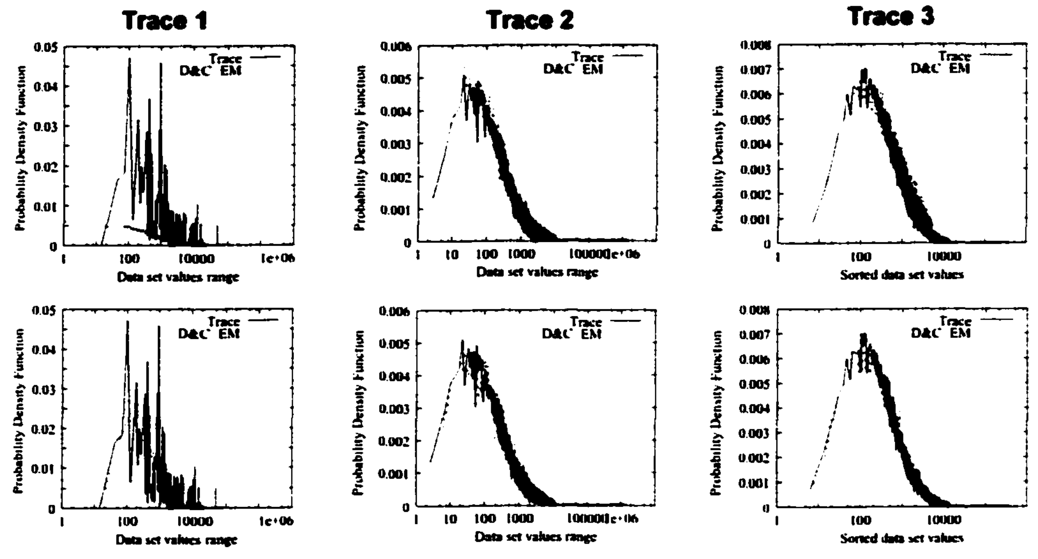

Figure 4.10: PDFs of the fittings for Traces 1, 2, and 3 using D&C EM (first row) and the refined D&C EM (second row).

and median matching for the refined D&C EM algorithm in Table 4.4. Observe that the means of the fitted models and data sets are better matched with the refined D&C EM (a direct outcome from better matching of the CDHs). There is no improvement in matching the second or any of the higher moments using the refined D&C EM, since we follow the same approach to capture the tail of the CDHs; the most significant property of the data sets under investigation. Note that matching the median is less accurate with the refined D&C EM algorithm.

	Data	D&C EM	Refined D&C EM
Trace 1			
Mean	4407.81	4393.56	4407.74
CV	7.28	7.86	7.80
Median	938.00	950.59	967.33
Trace 2			
Mean	6358.23	6164.50	6349.25
CV	5.87	5.13	5.04
Median	1081.14	1061.25	1268.19
Trace 3			
Mean	3459.86	3391.06	3441.82
CV	3.13	2.82	2.79
Median	1085.43	1086.59	1126.60

Table 4.4: Statistical evaluation of the D&C EM and refined D&C EM fittings.

The effect of the refined D&C EM algorithm is less obvious in the performance analysis of queueing systems. Because the first moment of the data sets are better matched, the average queue lengths of the queueing systems described in Subsection 4.2.2 are more accurate than what is shown in Figure 4.6 (first column). We stress that the accuracy of capturing the queue length distribution (body and tail) is maintained, emphasizing again that capturing the tail is critical for highly variable data sets.

4.3 D&C MM

We extend the idea of D&C EM by dividing the data set based on the expected value of each partition rather than the coefficient of variation. In this approach, we fit the data of each partition into PH distributions using methods of moment matching rather than the EM algorithm. This technique applies moment matching techniques in a divide and conquer fashion, hence we call it D&C MM.

Again, we base our technique on the analysis of the data set CDH. Contrary to D&C EM, D&C MM requires determining the number of partitions N before splitting the data into subsets. Partition boundaries are determined such that the expected value of each partition is M_d/N, where M_d is the expected value of the entire data set. Once the data set is partitioned, we compute the CV of each subset of data. If the CV of a subset of data is less than 1.0, i.e., the CV of the exponential distribution, we fit that subset of data into a hypoexponential distribution. If the CV is greater than 1.0, we fit that subset of data into a hyperexponential distribution. We fit each subset of data into a PH distribution using the Newton-Raphson method of moment matching, which is described in details in Appendix C and [98]. The resulting PH distribution is a mixture of hypoexponential and hyperexponential distributions. We formally present the D&C MM algorithm in Figure 4.11.

To illustrate the fitting methodology, we selected a data set containing the sizes of the requested files, i.e., the service process, measured during one entire day at the World Cup'98 Web site. We split the data set into four partitions and present in Figure 4.12 the PH distribution resulting form the merging of the four individual fittings; of those, the first one is a two-stage hyperexponential, while the other three are hypoexponential, with the last one very close to an Erlang (the numbers written

```
1. Build CDH from the data set.
     a  Compute expected value of the data set, $M_d$.
2. while (there are still CDH bins to be considered)
     a  include data of current bin into current partition $i$
     b  update expected value, $M_{Acc}$, of current partition
     c  if ($M_{Acc} > M_d/N$)
              Compute $CV$ of the partition
              if ($CV < 1$)
                 Use Newton-Raphson method to fit partition $i$ into a hypoexponential
              else
                 Use Newton-Raphson method to fit partition $i$ into a hyperexponential
              $M_{Acc} = 0$.
3. Generate final PH result
     Combine results of all PH fittings
```

Figure 4.11: D&C MM fitting algorithm.

inside each stage are the rates of the corresponding exponential distributions, while those on the arcs describe probabilistic splittings). To assess the quality of the overall PH fitting, we plot the PDF and CDF of the data and of our fit in Figure 4.13[4]. We also evaluate the accuracy of the fitting from the queueing system perspective, and present the results in Table 4.5 (we assume a M/PH/1 server with Poisson arrivals and the fitted PH distribution for service process). We conclude that D&C MM is a fast and accurate approach to fit data sets into PH distributions.

[4]Trace 1 in the analysis of D&C EM comes from the same server logs as the data set we use for analysis of D&C MM. The PDF of Trace 1 in Figure 4.5 is more jagged than the PDF presented in Figure 4.13 because in the former case we use shorter bins for our plots.

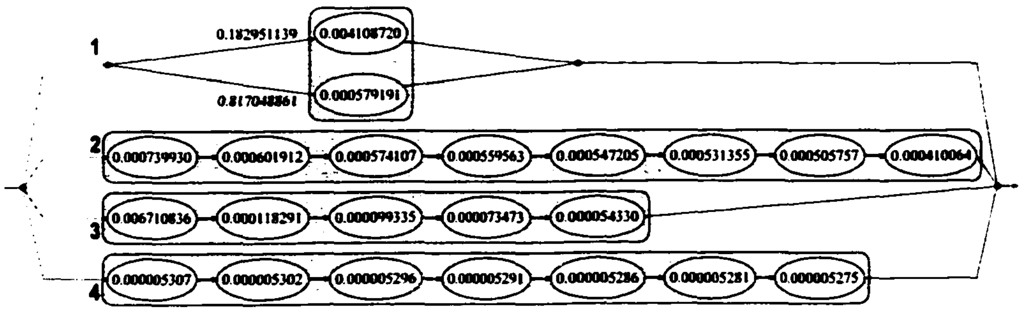

Figure 4.12: The resulting overall phase-type distribution.

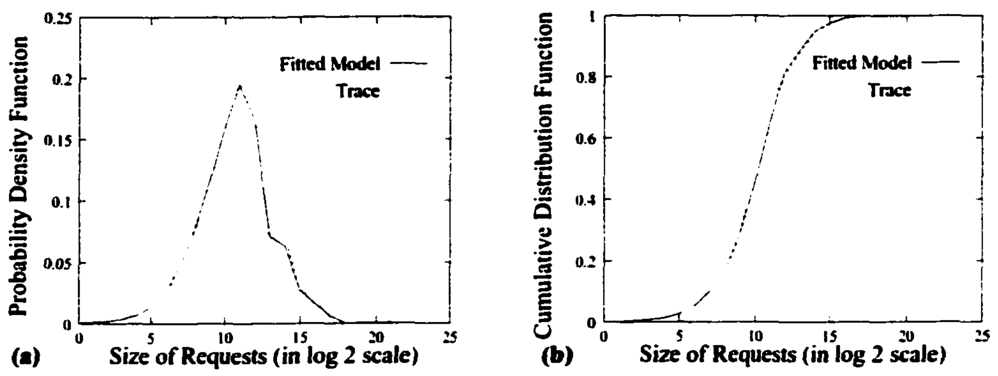

Figure 4.13: Comparing the empirical and fitted data: pdf (left) and CDF (right).

4.4 MAP fitting algorithm

In this section, we develop a technique for fitting highly correlated data into MAPs. The fitting technique that we propose is a hierarchical approach, based on the observation that a MAP essentially consists of a set of *control states* with arbitrary interactions among them, each representing an i.i.d. interarrival-time process. Each of the i.i.d. interarrival-time processes can exhibit long-tailed behavior, introducing additional states in the underlying Markov chain of the MAP process. We refer to these additional states as *phase-type states*.

We first employ standard Hidden Markov Model (HMM) methods to identify the

Arrival rate	Real data (trace-driven simulation)		Fitted data (analytical solution)	
	E[queue length]	E[slowdown]	E[queue length]	E[slowdown]
0.0000250	0.84	6.51	0.83	6.41
0.0000375	1.92	9.96	1.92	9.81
0.0000500	3.60	14.01	3.60	13.84
0.0000625	6.05	18.82	6.06	18.60
0.0000750	9.51	24.65	9.52	24.38
0.0000875	14.37	31.92	14.40	31.60
0.0001000	21.22	41.24	21.26	40.82
0.0001125	30.99	53.54	31.10	53.09
0.0001250	45.34	70.49	45.44	69.81
0.0001375	67.42	95.29	67.41	94.14
0.0001500	104.22	135.02	103.95	133.07

Table 4.5: Comparing the empirical and fitted data: performance results.

set of control states and define the interactions among these states, at a coarser time scale than the data set, in an attempt to capture the correlations among the data set points as well as some of the variability. Then we either use these results directly to construct an MMPP (recall that an MMPP is a special case of MAP, see Subsection 2.7), or we exploit these results together with an additional statistical analysis of the data set and the EM algorithm for fitting PH distributions in order to construct a more general MAP.

An overview of the basic steps of our hierarchical fitting methodology is provided in Figure 4.14. The algorithm input consists of the data set, few assumptions, and initial values (discussed in Subsection 4.4.1). We stress that our methodology does not place any restrictions on the structure of the underlying Markov chain of the MAP. The first step produces the set of outputs described in **1b** under the assumption of either exponential or hyperexponential sojourn time distributions for each control state. In the latter case, we have $\boldsymbol{\mu}_i = [\boldsymbol{\mu}_{i,1}, \ldots, \boldsymbol{\mu}_{i,m_H}]$ for each control state i, $1 \le i \le m_C$, where m_C denotes the number of control states and m_H denotes the number of phases in the hyperexponential distributions; otherwise, $\boldsymbol{\mu}_i$ is a scalar. The corresponding MMPP is constructed directly from the output of step **1**, whereas the remaining steps are used together with the output of step **1** to construct the corresponding MAP. In step **3**, the fitting of the data set sequence $\{S_i\}$ for control state i into a Coxian distribution uses a slightly modified version of an implementation of the EM algorithm [6, 72]. The remaining details of the basic steps in Figure 4.14 are explained in Subsections 4.4.1 and 4.4.2.

4.4.1 Hidden Markov model for parameter estimate

A *hidden Markov model* (HMM) with explicit state duration is a doubly stochastic process, whose intensity is controlled by a finite-state discrete-time Markov chain $\{J_n : n \in Z^+\}$ on the state space $\{i : 1 \le i \le m_C\}$ representing the set of control states. The time that has been spent in state J_n is denoted by τ_n, and the number of arrivals per time unit is denoted by r_n, which is the observable output associated with state J_n. It is usually assumed that the control states $\{J_n\}$ and the observations $\{r_n\}$ are conditionally independent with the conditional distribution of r_n dependent

```
1. Use HMM methods to construct the control states and their interactions
  a. Input
     - data set with $N$ data entries
     - desired number of control states
     - assumption for the arrivals per control state
        - Poisson
     - assumption for the sojourn times per control state
        - Exponential
        - Hyperexponential : desired number of phases
  b. Output
     - number of control states $m_C$
     - transition probability matrix $\mathbf{P}$ for the control states: $\mathbf{P} = [\mathbf{P}_{i,j}],\ 1 \le i,j \le m_C$
     - mean interarrival times $\boldsymbol{\lambda}_i^{-1}$ for $1 \le i \le m_C$:  $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{m_C}]$
     - (vector of) mean sojourn times $\boldsymbol{\mu}_i^{-1}$ for $1 \le i \le m_C$:  $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{m_C}]$
     - sojourn time probability vectors $\mathbf{p}_i^{\boldsymbol{\mu}}$ for $1 \le i \le m_C$, if using
       hyperexponential distribution:  $\mathbf{p}^{\boldsymbol{\mu}} = [\mathbf{p}_1^{\boldsymbol{\mu}}, \ldots, \mathbf{p}_{m_C}^{\boldsymbol{\mu}}]$;  $\mathbf{p}_i^{\boldsymbol{\mu}} = [\mathbf{p}_{i,1}^{\boldsymbol{\mu}}, \ldots, \mathbf{p}_{i,m_H}^{\boldsymbol{\mu}}]$
     - mapping $\{J_n\}$ of each data entry to its corresponding control state
2. Construct a data set sequence $\{S_i\}$ per control state $i$ using mapping
     $\{J_n\}$ from step 1 and the original data set
3. Feed each sequence $\{S_i\}$ to EM alg. to generate a Coxian
     distribution for the interarrival process of each control state
4. Compute the probability of state change upon arrival using mapping $\{J_n\}$
5. Construct $\mathbf{D}_0$ using:
     - transition probability matrix from step 1
     - model for sojourn times from step 1
     - Coxian model for interarrivals from step 3
6. Construct $\mathbf{D}_1$ using:
     - Coxian model for interarrivals from step 3
     - transition probability matrix from step 1
     - model for sojourn times from step 1
     - probability vector computed in step 4
```

Figure 4.14: Overview of our MAP parameter estimation methodology.

on J_n only.

Since this semi-Markov chain is not directly observable, the state sequence $\{J_n, \tau_n\}$ and the model parameters (i.e., the transition probability matrix $\mathbf{P}$ for the control states, the mean interarrival time $\boldsymbol{\lambda}_i^{-1}$, the vector of mean sojourn times $\boldsymbol{\mu}_i^{-1}$, the sojourn time probability vector $\mathbf{p}_i^{\mu}$ for each control state i, and the control state sequence $\{J_n\}$ of the data set) are estimated from the observed sequence $\{r_n\}$.

The main steps of the standard procedure for HMM with explicit state duration are summarized as follows:

- Given an initial set of assumptions for the HMM model parameters (e.g., based on an a posteriori knowledge of the empirical workload), obtain refined estimates of the model parameters by applying the HMM re-estimation algorithm with explicit state duration [74] to the given observation sequence $\{r_n\}$.
- Apply an HMM forward-backward algorithm with explicit state duration [74] to find the maximum a posteriori state estimate, J_n, for the given observation sequences $\{r_n\}$.

We refer the interested reader to [74] for an overview of the details on these standard HMM algorithms. Additional technical details can be found in [97, 108, 110] and the references cited therein.

Following the above procedure, we can obtain the maximum likelihood model parameters for the given observation sequence $\{r_n\}$ and the state space $\{1, \ldots, m_C\}$. Let $\widehat{H}_i(\tau)$ denote the estimated non-parametric probability mass function for the sojourn time τ of state i, and let $\widehat{O}_i(r)$ denote the estimated non-parametric probability mass function for the observation r of state i.

The total number of model parameters can be further reduced if the observation distribution or the state sojourn time distribution is approximated by some parametric distributions such as Gaussian, Poisson or gamma distributions [91, 62]. In this case, one only needs to estimate a few parameters that specify the selected distribution functions. Ferguson [31] has shown that the parameters for the parametric sojourn time distribution $H_i(\tau)$ and the parametric observation distribution $O_i(r)$ for state i can be found by maximizing $\sum_\tau \widehat{H}_i(\tau)\ln(H_i(\tau))$ and $\sum_r \widehat{O}_i(r)\ln(O_i(r))$ subject to the stochastic constraints $\sum_\tau H_i(\tau) = 1$ and $\sum_r O_i(r) = 1$.

If the arrival process for each control state is Poisson and the control state sojourn times follow a hyperexponential distribution, i.e.,

$$O_i(r) = \frac{(\lambda_i t)^r}{r!} e^{-\lambda_i t}, \tag{4.2}$$

$$H_i(\tau) = \sum_{j=1}^{m_H} \mathbf{p}_{ij}^{\boldsymbol{\mu}} \, \boldsymbol{\mu}_{ij} \, e^{-\boldsymbol{\mu}_{ij}(\tau-1)}, \tag{4.3}$$

where $\lambda_i \geq 0$, $\boldsymbol{\mu}_{ij} \geq 0$ and $\sum_j \mathbf{p}_{ij}^{\boldsymbol{\mu}} = 1$, then the arrival rate λ_i of the Poisson process for state i can be estimated by $\widehat{\lambda}_i = \sum_r \widehat{O}_i(r) r$ [31, 91]. The parameters $\mathbf{p}_{ij}^{\boldsymbol{\mu}}$ and $\boldsymbol{\mu}_{ij}$ of the hyperexponential distribution can be determined numerically via Eq. (4.3).

Finally, as part of the initialization step, we set the total number of control states to a pre-specified input parameter (which is analyzed in our experiments). If this parameter is a sufficiently large integer, in the re-estimation procedure, we delete the states that are never visited. As such the value of m_C is reduced to a smaller number of control states. This led to a maximum of 20 control states for the data sets used in our study. We also need to initialize the elements of the transition probability matrix, the control state sojourn time distributions, and the initial control state probability vector. An often used choice is to assume that these initial values for the model parameters are uniformly distributed. In addition, we assume the initial values of the control-state arrival rates to be proportional to the state index, i.e., $\lambda_i = r_{max}\frac{i}{m_C}$ where i is the index of the control state, r_{max} is the maximum value of r, and m_C is the total number of control states.

4.4.2 Generation of MAP from HMM output

The above HMM methods produce the output described in Figure 4.14, which includes the sequence $\{J_n\}$, $1 \leq n \leq N$, representing the mapping of the entries in the data set to the set of control states. That is, as part of the HMM analysis, each interarrival time in the data set is assigned to one of the m_C control states. We first construct a new data set sequence $\{S_i\}$, $1 \leq i \leq m_C$, that consists of all of the interarrival times from the data set with $J_n = i$, in the same relative order as in the original data set. We also compute from the sequence $\{J_n\}$, $1 \leq n \leq N$, the probability vector $\widehat{\mathbf{p}}$ of dimension m_C, where the element $\widehat{\mathbf{p}}_i$ denotes the probability that upon an arrival from control state i the process switches to another control state. Specifically, we have

$$\widehat{\mathbf{p}}_i = \frac{\sum_{j=2}^{N} I_{J_j \neq i, J_{j-1}=i}}{\sum_{j=2}^{N} I_{J_j = i, J_{j-1}=i} + \sum_{j=2}^{N} I_{J_j \neq i, J_{j-1}=i}}, \tag{4.4}$$

where I_A denotes the indicator function for event A having the value 1 if A occurs, and the value 0 otherwise. Thus, with probability $1 - \widehat{\mathbf{p}}_i$ the process immediately returns to the same control state i.

To help facilitate the description of the procedure for generating MAPs from the above set of variables, we need the following definitions:

- $\mathbf{\Lambda} = \textbf{diag}(\lambda_1, \ldots, \lambda_{m_C})$ is a diagonal matrix of order m_C whose diagonal elements are the elements of vector $\boldsymbol{\lambda}$;
- $\mathbf{P}^{\boldsymbol{\mu}}_i$ is the $m_H \times m_H$ matrix whose rows are all equal to $\mathbf{p}^{\mu}_i$, $1 \leq i \leq m_C$;
- $\mathbf{\Phi} = \textbf{diag}(\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_{m_C})$ is a (block) diagonal matrix of order $m_C \cdot m_H$, where $\mathbf{\Phi}_i = \textbf{diag}(\boldsymbol{\mu}_{i,1}, \ldots, \boldsymbol{\mu}_{i,m_H})$ is a diagonal matrix of order m_H;
- $\mathbf{I}_k$ is the order k identity matrix;
- $\textbf{col}(M, k, j)$ is a matrix function that partitions the columns of the matrix M into blocks of size k and then extracts the j^{th} such block of columns of size k from matrix M (the resulting matrix has the same number of rows as M and k columns);
- $\textbf{row}(M, k, j)$ is a matrix function that partitions the rows of the matrix M into blocks of size k and then extracts the j^{th} such block of rows of size k from matrix M (the resulting matrix has k rows and the same number of columns as M);
- $\mathbf{V} = [\mathbf{V}_1 \, \mathbf{V}_2 \, \ldots \, \mathbf{V}_{m_C}]$, where $\mathbf{V}_i = \textbf{col}(\mathbf{P}, 1, i) \otimes \mathbf{P}^{\boldsymbol{\mu}}_i$, $1 \leq i \leq m_C$.

The off-diagonal elements of the matrix $\mathbf{D}_0^{\text{MMPP}}$ and the matrix $\mathbf{D}_1^{\text{MMPP}}$ for an MMPP with exponential interarrival times and hyperexponential sojourn times per control state can be expressed as

$$\mathbf{D}_0^{\text{MMPP}} = \mathbf{\Phi V}, \qquad \mathbf{D}_1^{\text{MMPP}} = \mathbf{\Lambda} \otimes \mathbf{I}_{m_H}. \tag{4.5}$$

Then the diagonal element of each row of $\mathbf{D}_0^{\text{MMPP}}$ is computed as the negative sum of the non-diagonal elements on the same row in the matrix $\mathbf{D}_0^{\text{MMPP}} + \mathbf{D}_1^{\text{MMPP}}$. The number of states in this MMPP is $m_C \cdot m_H$. During the numerical experiments, we observed that some of the values in vectors $\mathbf{p}_i^{\boldsymbol{\mu}}$, $1 \leq i \leq m_C$, are very small, i.e., less than a desired tolerance of accuracy. The states that are represented by these probabilities are removed from the set of states during the generation of $\mathbf{D}_0$ and $\mathbf{D}_1$ to avoid numerical problems in the solution of the MAP/PH/1 queues. We note that the size of the state space is decreased by up to 30% with this simple state reduction technique.

From the HMM output, we can also construct a more general MAP by using the same matrix $\mathbf{D}_0$ from the above MMPP and modifying the matrix $\mathbf{D}_1$ by making use of the probability vector $\widehat{\mathbf{p}}$. Define $\widehat{\mathbf{P}} = \mathbf{diag}(\widehat{\mathbf{p}}_1, \ldots, \widehat{\mathbf{p}}_{m_C})$ to be the order m_C diagonal matrix corresponding to the vector $\widehat{\mathbf{p}}$. We then have

$$\begin{aligned}
\mathbf{D}_0^{\text{MAP}} &= \mathbf{D}_0^{\text{MMPP}}, \\
\mathbf{D}_1^{\text{MAP}} &= \left(\mathbf{I}_{m_C \cdot m_H} - \mathbf{diag}(((\widehat{\mathbf{P}} \otimes \mathbf{I}_{m_H})\mathbf{Ve})^T) + (\widehat{\mathbf{P}} \otimes \mathbf{I}_{m_H})\mathbf{V}\right) \mathbf{D}_1^{\text{MMPP}}.
\end{aligned}$$

We compute from the sequence $\{J_n\}$, $1 \leq n \leq N$, only the probability of leaving a control state upon arrival. The probability of reaching any other control state is not computed from this sequence, but rather from the probability transition matrix **V** since it is estimated using a similar analysis (which is reflected in the definition of matrix $\mathbf{D}_1^{\text{MAP}}$ in Eq.(4.6)).

Another set of MAP processes is obtained by incorporating the results of fitting the data in each of the data set sequences $\{\mathcal{S}_i\}$, $1 \leq i \leq m_C$, to a Coxian distribution using the EM algorithm.[6] This computes for each control state i the vector $\hat{\boldsymbol{\alpha}}_i$ and the matrix $\hat{\mathbf{T}}_i$, $1 \leq i \leq m_C$, both of order m_X. We define

$$\begin{aligned}
\mathbf{U}_i &= \mathbf{row}(\mathbf{D}_0^{\text{MAP}}, m_H, i) \otimes \hat{\mathbf{T}}^i, \\
\mathbf{X}_i &= \mathbf{row}(\mathbf{I}_{m_C m_H} - \mathbf{diag}(((\hat{\mathbf{P}} \otimes \mathbf{I}_{m_H})\mathbf{V}\mathbf{e})^T) + (\hat{\mathbf{P}} \otimes \mathbf{I}_{m_H})\mathbf{V}, m_H, i) \otimes \hat{\mathbf{T}}_i^o \hat{\boldsymbol{\alpha}}_i \\
\mathbf{U} &= [\mathbf{U}_1^T, \cdots, \mathbf{U}_{m_C}^T]^T, \qquad \mathbf{X} = [\mathbf{X}_1^T, \cdots, \mathbf{X}_{m_C}^T]^T.
\end{aligned}$$

Then the matrices $\mathbf{D}_0^{\text{MAP-C}}$ and $\mathbf{D}_1^{\text{MAP-C}}$, where MAP-C stands for the MAP with Coxian interarrival processes for each control state, can be expressed as follows

$$\mathbf{D}_0^{\text{MAP-C}} = \mathbf{U}, \qquad \mathbf{D}_1^{\text{MAP-C}} = \mathbf{X}. \tag{4.6}$$

The total number of states for this MAP is $m_C \cdot m_H \cdot m_X$. In the same manner as described above, the states with probabilities that are very close to 0 are removed from the final version of the MAP.

4.4.3 Experimental results

Our objective in proposing the algorithm of Figure 4.14 is to be able to fit data sets that exhibit long-range dependence into MAPs and consequently analyze the performance of queueing systems that operate under such correlated arrival processes. Analysis of data measured at Web servers show that the request arrivals at a Web server are correlated, while the service process is i.i.d. [105]. Further, we assume that the arrival and the service process are mutually independent. Hence, we model a Web server as a MAP/PH/1 queue and investigate the accuracy of the algorithm of Figure 4.14 only from the queueing system's perspective. Appendix B describes how to generate the MAP/PH/1 queueing system once the parameters of the MAP and PH distributions are known.

4.4.3.1 Traces

We use Web servers measurement data for our experiments. Each access log contains the time epoch of the n^{th} request and the number of bytes comprising the n^{th} request. The unit of time in the access logs available to us is one second, which is quite standard. Since there are typically tens or even hundreds of requests within a second, we use the method developed in [111] to provide a finner arrivals time scale. The coarser time granularity directly provides us with the discrete-time batch process for the number of client requests per second.

We have selected three representative data sets to be used in our experimental analysis. Each of the data sets exhibit correlation, whose indicator is the value of the *Hurst parameter* [10]. A data set with Hurst parameter higher than 0.5 is said to be long-range dependent. The first data set, denoted as *Trace A*, represents the peak traffic period for one of the Web sites of interest. This data set is long-range dependent with a Hurst parameter H_A of approximately 0.78. The second data set, *Trace B*, represents the off-peak traffic period for one of the Web sites, which is long-range dependent with a Hurst parameter $H_B \approx 0.64$. The third data set, *Trace C*, is somewhat artificial but included here to represent a more extreme case where the Hurst parameter H_C is around 0.9. Each of these data sets consist of traffic periods whose lengths are on the order of five hours and consist of more than 500,000 data points.

4.4.3.2 Experimental setting

We consider fitting the interarrival times of each data set with a wide variety of models that can be obtained from our methodology under the following assumptions:

- an exponential (Exp) or x-phase Coxian (Cox$x$) distribution for the interarrival times associated with each control state;
- an exponential or x-phase hyperexponential (Hrx) distribution for the sojourn times of each control state.

To facilitate the presentation of results, we shall use the notation MMPP(s,d) and MAP(s,a,d) where s denotes the number of control states, d denotes the distribution of sojourn times for each control state, and a denotes the distribution that models the interarrival times associated with each control state.

In order to evaluate the accuracy of our approach, we compare various steady-state performance measures of the MAP/PH/1 queue against the corresponding measures obtained via trace-driven simulations. We assume that requests are served in a FCFS manner, and that the server depletes work at rate C, where C is a deterministic constant. By varying the parameter C, different server loads are obtained. The value of C can only be reduced up to points that still maintain a stable regenerative G/G/1 queue (in the sense that the system empties with probability 1 and that there are a sufficiently large number of such regeneration points).

4.4.3.3 Peak Traffic Period (Trace A)

We vary the number of control states used by the HMM algorithm as part of our methodology for fitting the interarrival times of Trace A to various MAPs. This makes it possible for us to examine the impact of the size of the underlying Markov chain on the accuracy of the model fitting. We start with 2 control states, and then increase to 5 and 10 states. The mean response times under a small subset of these MAPs as a function of the traffic intensity ρ are plotted in Figure 4.15(a), together with the corresponding simulation results for Trace A. Our results clearly demonstrate that the accuracy of the fitting improves significantly with increases in the number of control states, as expected. (Note that the MAP(5,Exp,Exp) results are somewhat more accurate than the MAP(2,Exp,Exp) results, which are not shown.) This is because more control states in the underlying Markov chain provide greater flexibility which makes it possible to better capture not only the dependence structure but also the variability of the arrival process. The output of the HMM algorithm for larger numbers of control states exhibit small probabilities for entering a few of the control states, together with small transition rates for leaving these control states once entered. This suggests that such control states improve the ability of the MAP to capture the tail of the interarrival process, and that the degree to which this is possible improves with increases in the number of control states.

In order to isolate, to some extent, the impact of the dependence structure on mean response time measures, we have ignored such dependencies and fitted the interarrival times of Trace A to a PH distribution. The mean response time measures for this PH/PH/1 queue are also provided in Figure 4.15(a). It can be clearly observed from these results that the PH/PH/1 fitting is poor and, with the exception of light traffic intensities, the PH/PH/1 queue is simply not capable of capturing the performance of the queueing system under Trace A. Conversely, the MAP models that capture the dependence structure do a much better job of matching the queueing system performance, particularly at heavier loads, where the accuracy increases with the complexity of the MAP.

In a similar manner, our methodology is used for fitting Trace A to various MMPPs, and the corresponding mean response times under a small subset of these

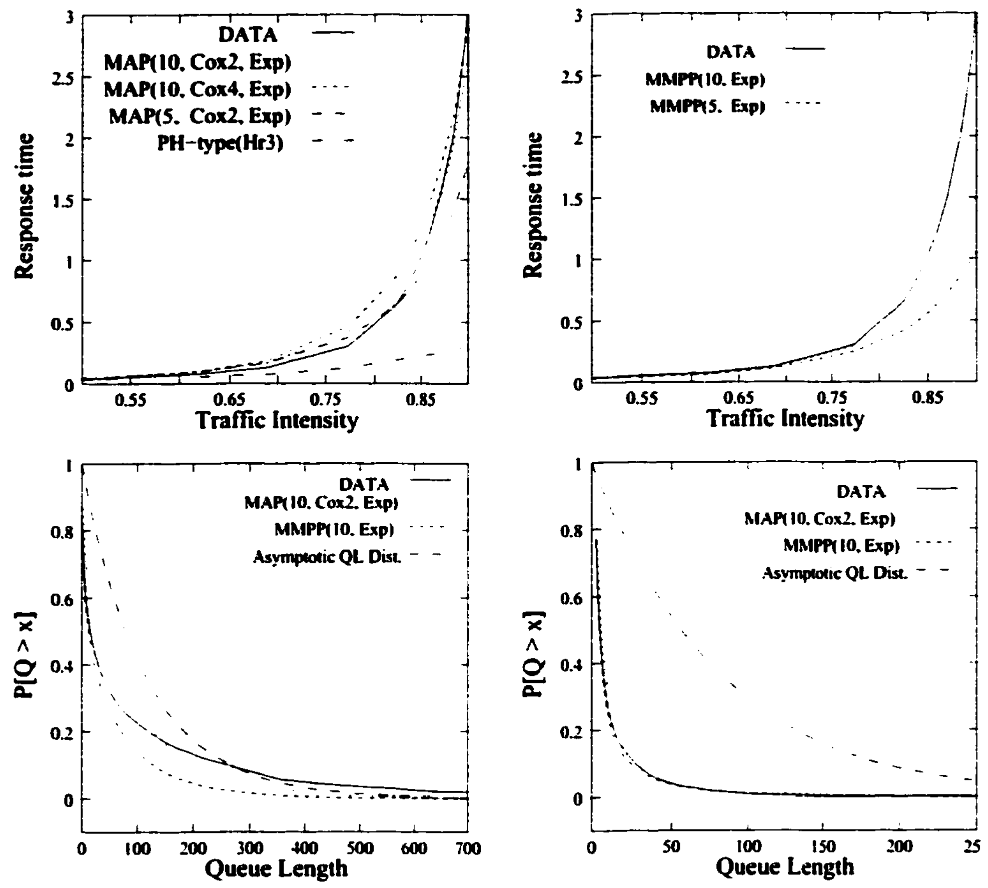

Figure 4.15: Response time as function of traffic intensity for (a) MAP models, (b) MMPP models and queue length tail distribution of fitted models for (c) moderate, (d) high system loads for Trace A

MMPPs are plotted in Figure 4.15(b). The results from simulation are also included in the figure for comparative purposes. We continue to observe that the larger the number of control states, the more accurate the fitting. Once again, more control states in the underlying Markov chain provide greater flexibility that makes it possible to better capture both the dependence structure and the variability of the arrival process, for the reasons described above.

We observe that one of the MMPP models provides the most accurate results in comparison with simulation for light loads. Under heavier loads, however, one of the MAPs tends to provide the most accurate results. Specifically, MAP(10,Cox2,Exp) provides the best accuracy with a relative error always less than 20%. We further observe that the models using Coxian distributions for the interarrival time process of each control state tend to provide better fits. Based on a simple statistical analysis of the sequence $\{J_n\}$ defined in Subsection 4.4.1, we further observe that the interarrivals for each control state are not exponential, which is why we use the Coxian distribution to model each of these control-state interarrival processes.

We also study the tail behavior of the queue length distribution, using Eq.(3.5), for the best MAP and MMPP models, i.e., MAP(10,Cox2,Exp) and MMPP(10,Exp). The asymptotic queue length tail distributions corresponding to MAP(10,Cox2,Exp) model, based on the caudal characteristic η via Eq.(3.11), and the tail of the queue length distribution from the direct simulation of Trace A are used for comparison. Figure 4.15(c) plots these five queue length tail distributions for the traffic intensity $\rho = 0.77$, which represents a case where the system is moderately loaded. The corresponding set of results for a traffic intensity of $\rho = 0.90$, which represents a heavily loaded system, are presented in Figure 4.15(d). Note that in each of these figures we only plot the asymptotic queue length tail distribution for the MAP model because the corresponding curve for the MMPP model is quite close to that of the MAP model.

We observe that the queue length tail distributions obtained under the MAP and

MMPP models provide a reasonably close match with the corresponding tail distribution obtained from simulation over a relatively wide range. In the case of moderate load, i.e., $\rho = 0.77$, the tail distribution from the MMPP model closely matches the simulation results for small queue length values, which helps to explain the low relative error values at light to moderate loads. Conversely, the tail distribution from the MAP model overestimates the simulation results for all but very large queue length values. This causes the MAP to yield poorer relative errors at light to moderate loads, although still always less than 20%. In the case of heavier load, i.e., $\rho = 0.90$, the tail distribution from the MMPP model continues to provide a close match with the simulation results but only for very small queue length values. The tail distribution from the MAP model continues to overestimate the simulation results over a relatively large range of queue length values, crossing at around a queue length of 140. In fact, the accuracy of the expected response time under the MAP model is achieved in part by pushing this crossover point relatively far to the right (to larger queue lengths). This further explains why the expected response times under the MAP model underestimate those obtained from simulation at heavier loads. More accurate response times under the MAP model at heavier loads can be obtained by increasing the number of (control and/or phase-type) states in the underlying Markov chain.

We also observe that the asymptotic queue length tail distribution based on the caudal characteristic η dominates all other tail distributions, for both $\rho = 0.77$ and

$\rho = 0.90$, across a relatively wide range of queue length values. The tail of the queue length distribution from simulation eventually crosses the asymptotic tail distribution, as expected due to the dependence structure and variability in Trace A, but these crossover points occur at queue length values greater than 250 which can be considered to be a relatively high value of queue length. We note that the maximum response time value plotted for the MAP(10,Cox2,Exp) model represents $\rho = 0.9$, $\widehat{\rho} = 0.94$ and $\eta = 0.99$. This illustrates and quantifies how the latter two variables provide a better measure of effective system load than the standard traffic intensity ρ.

4.4.3.4 Off-Peak Traffic Period (Trace B)

The same set of experiments and analysis as those described in Subsection 4.4.3.3 are performed on Trace B. Since the results in Subsection 4.4.3.3 demonstrate that a model with 10 control states fits the interarrival process much more accurately than the corresponding models with fewer control states, here we focus solely on models with 10 control states in the analysis of Trace B.

Based on Eqs.(4.5), (4.6), (4.6), and (4.6), several different MAP and MMPP models are fitted to the interarrival process of Trace B. The mean response times under these MAPs and MMPPs as a function of the traffic intensity ρ are respectively plotted in Figures 4.16(a) and 4.16(b). In Figure 4.16(a) we observe that the PH/PH/1 model does not perform much worse than some of the MAP and MMPP models. This is directly related to the weaker long-range dependence of the interarrival process of Trace *B*. From among the set of MAP models, MAP(10,Cox2,Exp) performs the best with a relative error always less than 12%. On the other hand, the MMPP(10,Exp) model performs slightly better with a worst case relative error of 10%. This supports the notion that the MMPP is a good model for data sets which have a relatively weak long-range dependence structure, or a short-range dependence structure.

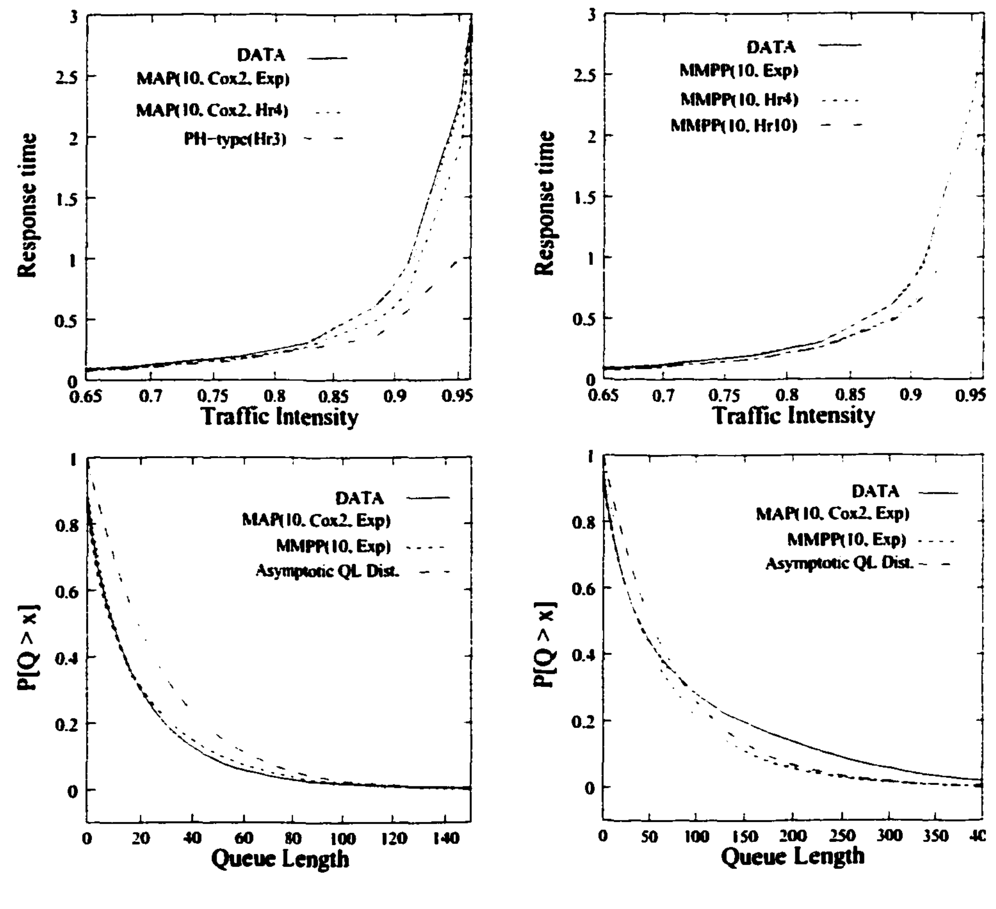

Figure 4.16: Response time as function of traffic intensity for (a) MAP models, (b) MMPP models and queue length tail distribution of fitted models for (c) moderate, (d) high system loads for Trace B

The queue length tail distributions for the models MMPP(10,Exp) and MAP(10,Cox2,Exp) as well as the asymptotic behavior (as characterized by Eq.(3.11) in terms of the caudal characteristic η) for model MAP(10,Cox2,Exp), are compared with the queue length tail distribution obtained from the simulation of Trace B in Figure 4.16(c) for a traffic intensity of 0.88. The same curves for traffic intensity of 0.95 are shown in Figure 4.16(d). These plots illustrate that for both moderate and relatively heavy traffic intensities MMPP(10,Exp) is a slightly better fit than MAP(10,Cox2,Exp). However, for heavier traffic intensities, the MAP(10,Cox2,Exp) curve follows the simulation curve for larger values of queue length than does the MMPP(10,Exp) curve. Note that the maximum response time value plotted for the MAP(10,Cox2,Exp) model represents $\rho = 0.97$, $\hat{\rho} = 0.99$ and $\eta = 0.99$.

4.4.3.5 Strong Long-Range Dependence (Trace C)

We perform the same set of experiments and analysis on the Trace C data set as those considered in Subsections 4.4.3.3 and 4.4.3.4. Figure 4.17(a) presents the mean response time as a function of the traffic intensity for all of the fitted MAP models compared against the simulation curve for Trace C. We note that the MAPs do a very good job of accurately capturing the queueing system performance even though the dependence structure of Trace C is much stronger than that found in Traces A and B. It is this strong long-range dependence in Trace C that causes the MAP models with hyperexponential sojourn time distributions for the control states to perform better than the cases where the sojourn times are assumed to be exponentially distributed.

Moreover, unlike the 2-phase Coxian distributions that are used to fit the interarrival process for each of the control states for Traces A and B, we choose a 4-phase Coxian distribution for fitting the interarrival process of each control state for Trace C. These 4-phase Coxian distributions are essentially Erlang distributions because we find that the coefficient of variation for each of the constructed control-state trace sequences (see Subsection 4.4.2) is less than 0.5.

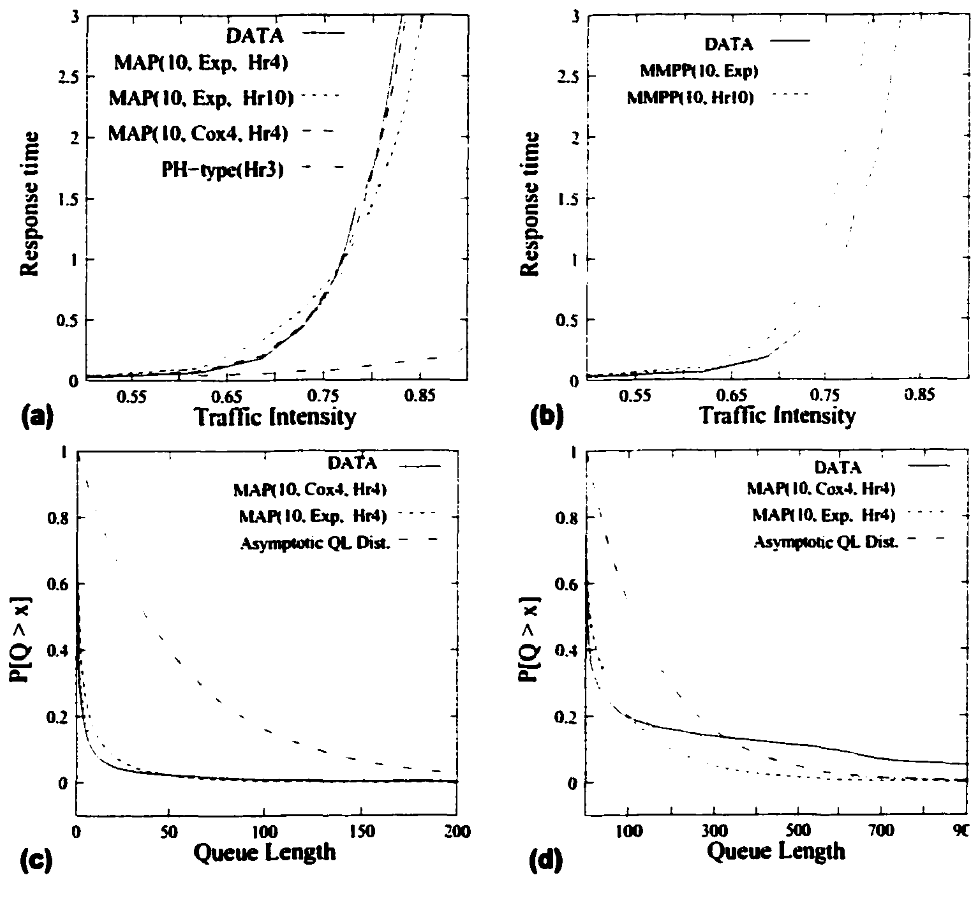

Figure 4.17: Response time as function of traffic intensity for (a) MAP models, (b) MMPP models and queue length tail distribution of fitted models for (c) moderate, (d) high system loads for Trace C

Figure 4.17(b) is similar to Figure 4.17(a) but for all of the fitted MMPP models and shows that the MMPP models only roughly approximate the simulation curve. The best fit for Trace C is provided by the MAP(10,Cox4,Hr4) model with relative error less than 12%.

The queue length tail distributions for the two best fitting models, MAP(10,Cox4,Hr4) and MAP(10,Exp,Hr4), are plotted in Figure 4.17(c) for the moderate traffic intensity of 0.69, and in Figure 4.17(d) for the high traffic intensity of 0.83. These plots illustrate that the tail of the queue length distribution for Trace C obtained by simulation is heavy and that the MAP(10,Cox4,Hr4) model does a much better job of capturing the characteristics of this heavy tail up to a relatively large queue length value under both moderate and high traffic intensities. However, at high traffic intensities, the heavier tail of the simulation results eventually crosses from below that of the MAP(10,Cox4,Hr4) model, beyond which it decays much more slowly than all other tail distribution curves. This further explains why the expected response times under the MAP model underestimate those obtained from simulation at heavier loads. On the other hand, as previously noted, the range of traffic intensities that cover mean response time values from ES to $100 \times$ ES represent by far the set of traffic intensities that might be of interest in practice. We note that the maximum response time value plotted for the MAP(10,Cox4,Hr4) model represents $\rho = 0.83$, $\hat{\rho} = 0.84$ and $\eta = 0.99$. This illustrates and quantifies how the latter two variables provide a better measure of effective system load than the standard traffic intensity ρ.

4.5 Chapter summary

In this chapter, we proposed three new techniques for fitting data sets that exhibit high variability and long-range dependence into PH distributions and MAPs, respectively. We presented D&C EM, a divide-and-conquer implementation of the EM algorithm, for fitting highly variable data sets into hyperexponential and more general PH distributions. We demonstrated via experimental results that D&C EM provides fits that accurately match the major statistical properties of the data sets such as median, first, second, and third moments. We tested the accuracy of D&C EM from the queueing system's perspective as well. The queueing systems, that used as service process the D&C EM fits, captured closely the behavior of the systems, matching the behavior evaluated via trace-driven simulations. We showed that, as a fitting technique, D&C EM assures both accuracy and efficiency. Using a similar approach as in D&C EM, we also developed D&C MM, which fits data sets into PH distributions by partitioning them, and using the method of moment matching to fit each partition into a PH distribution. Apart from D&C EM and D&C MM, we proposed a technique for fitting correlated data sets into MAPs. The technique uses basic knowledge of Hidden Markov models, statistical analysis, and the EM algorithm to generate a MAP. We tested the accuracy of the technique from the queueing system's perspective via numerous experiments using Web server measurement data.

Chapter 5

ETAQA Methodology

In this chapter, we outline ETAQA, the methodology that we propose for solving and computing various measures of interests for GI/M/1-, M/G/1-type processes and their intersection, i.e., QBD processes. ETAQA stands for Efficient Technique for Analyzing QBD processes by Aggregation. Although we present ETAQA as a generalized methodology, it started as a technique to solve a special case of QBD processes [24]. ETAQA has a simple formalization, it is computationally efficient, numerically exact, yet provides enough information to conduct detailed analysis of the given process.

The traditional solution algorithms, described in Chapter 3, compute the stationary probability vector with a recursive function based on $\mathbf{G}$ (for the case of M/G/1-type processes) or $\mathbf{R}$ (for the case of GI/M/1-type processes), and iterative procedures are used for determining $\mathbf{G}$ or $\mathbf{R}$. Alternative algorithms for the computation of $\mathbf{G}$ or $\mathbf{R}$ have been proposed (e.g., the work of Latouche [46] for the efficient computation of $\mathbf{R}$ and of Meini [60] for the efficient computation of $\mathbf{G}$).

Distinctively from the classic techniques of solving M/G/1-type and GI/M/1-type processes, we recast the problem into solving a finite system of $m+2n$ linear equations, where m is the number of states in the boundary portion of the process and n is the number of states in each of the repetitive "levels". The proposed methodology uses basic, well-known results for Markov chains. Assuming that the state space $\mathcal{S}$ is partitioned into sets $\mathcal{S}^{(j)}$, $j \geq 0$, instead of evaluating the probability distribution of *all* states in each $\mathcal{S}^{(j)}$, we calculate the *aggregate* probability distribution of n classes of states $\mathcal{T}^{(i)}$, $1 \leq i \leq n$, appropriately defined (see Figure 5.1).

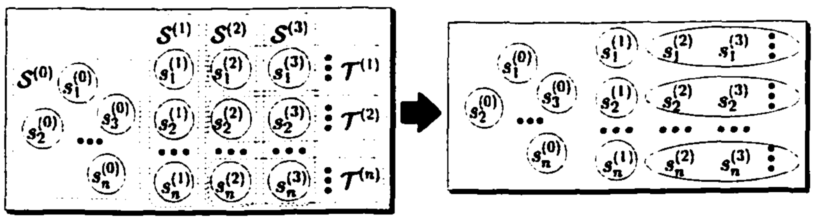
$\mathcal{S}^{(0)}$
$\mathcal{S}^{(1)}$
$\mathcal{S}^{(2)}$
$\mathcal{S}^{(3)}$
$\mathcal{T}^{(1)}$
$\mathcal{T}^{(2)}$
$\mathcal{T}^{(n)}$
$s_1^{(0)}$ $s_2^{(0)}$ $s_3^{(0)}$ $s_n^{(0)}$
$s_1^{(1)}$ $s_2^{(1)}$ $s_n^{(1)}$
$s_1^{(2)}$ $s_2^{(2)}$ $s_n^{(2)}$
$s_1^{(3)}$ $s_2^{(3)}$ $s_n^{(3)}$

Figure 5.1: Aggregation of an infinite $\mathcal{S}$ into a finite number of states.

We note that our approach does not require any restriction on the form of the chain's repeating pattern, thus can be applied to *any* type of M/G/1, GI/M/1, or QBD chain. The proposed methodology is both efficient and exact, but also numerically stable. ETAQA does not require the calculation of steady state probabilities in explicit recursive form, yet it provides the means for calculating a rich set of measures of interest such as the expected queue length and any of its higher moments. Detailed comparisons with the traditional methods show that the proposed methodology results in significantly more efficient solutions for the case of M/G/1-type and QBD processes. For the case of GI/M/1-type processes, our methodology exhibits the same complexity as the traditional one for the computation of the stationary probability vector, but it results in more complex formulas when computing measures of interest.

This chapter is organized as follows. In Section 5.1, we present ETAQA for M/G/1-type processes, derive the computation of measures of interest, and analyze the complexity of the method with respect to both storage and computation requirements. Section 5.2 outlines ETAQA for GI/M/1-type processes, its computation of measures of interest, and its complexity analysis. In Section 5.3. we present ETAQA for QBD processes, show how to compute measures of interest, and conclude with its complexity analysis. Section 5.4 presents experimental results that demonstrate the computational efficiency of ETAQA for solution of M/G/1-type processes. In Section 5.5, we evaluate the numerical stability of ETAQA for M/G/1-type processes by comparing its performance with numerical stable matrix-analytic algorithms. In Section 5.6, we describe MAMSOLVER, a matrix-analytic methods tool that provides software implementations for ETAQA as well as the other existing state-of-the-art algorithms for the solution of M/G/1-type, GI/M/1-type and QBD processes. We conclude the chapter with a summary of the results.

5.1 ETAQA-M/G/1

In Subsection 3.4 we outlined the matrix analytic method for the solution of M/G/1-type processes. Here we introduce an aggregated technique that computes only $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ and the aggregated stationary probabilities of n classes of states. The first step toward the solution of an M/G/1-type process is the computation of matrix $\mathbf{G}$. We assume that $\mathbf{G}$ is available, i.e., it has been computed using an efficient iterative method, e.g., the cyclic reduction algorithm [12], or that it can be explicitly obtained [78].

Recall that, as defined in Eq.(2.22), for the case of M/G/1-type processes the block-partitioned infinitesimal generator has the following form:

$$\mathbf{Q}_{M/G/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} & \widehat{\mathbf{F}}^{(2)} & \widehat{\mathbf{F}}^{(3)} & \widehat{\mathbf{F}}^{(4)} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The block partitioning of the infinitesimal generator defines a block partitioning of the stationary probability vector $\boldsymbol{\pi}$ as $\boldsymbol{\pi} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, ...]$ with $\boldsymbol{\pi}^{(0)} \in \mathbb{R}^m$ and $\boldsymbol{\pi}^{(i)} \in \mathbb{R}^n$, for $i \geq 1$. Furthermore, we can rewrite the matrix equality $\boldsymbol{\pi} \cdot \mathbf{Q}_{M/G/1} = \mathbf{0}$ as:

$$\left\{ \begin{array}{ll} \boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{L}} + \boldsymbol{\pi}^{(1)}\cdot\widehat{\mathbf{B}} & = \mathbf{0} \\ \boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{F}}^{(1)} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{B} & = \mathbf{0} \\ \boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{F}}^{(2)} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{F}^{(1)} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(3)}\cdot\mathbf{B} & = \mathbf{0} \\ \boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{F}}^{(3)} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{F}^{(2)} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{F}^{(1)} + \boldsymbol{\pi}^{(3)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(4)}\cdot\mathbf{B} & = \mathbf{0} \\ \vdots & \end{array} \right. . \tag{5.1}$$

Theorem Given an ergodic CTMC with infinitesimal generator $\mathbf{Q}_{M/G/1}$ having the structure shown in Eq.(2.22), with stationary probability vector $\boldsymbol{\pi} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, ...]$, the system of linear equations

$$\mathbf{x} \cdot \mathbf{X} = [1, \mathbf{0}], \tag{5.2}$$

where $\mathbf{X} \in \mathbf{R}^{(m+2n)\times(m+2n)}$ is defined as follows

$$\mathbf{X} = \left[\begin{array}{c|c|c|c} \mathbf{1}^T & \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} - \sum_{i=3}^{\infty}\widehat{\mathbf{S}}^{(i)}\cdot\mathbf{G} & (\sum_{i=2}^{\infty}\widehat{\mathbf{F}}^{(i)} + \sum_{i=3}^{\infty}\widehat{\mathbf{S}}^{(i)}\cdot\mathbf{G})^{\circ} \\ \mathbf{1}^T & \widehat{\mathbf{B}} & \mathbf{L} - \sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G} & (\sum_{i=1}^{\infty}\mathbf{F}^{(i)} + \sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G})^{\circ} \\ \mathbf{1}^T & \mathbf{0} & \mathbf{B} - \sum_{i=1}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G} & (\sum_{i=1}^{\infty}\mathbf{F}^{(i)} + \mathbf{L} + \sum_{i=1}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G})^{\circ} \end{array}\right], \tag{5.3}$$

admits a unique solution $\mathbf{x} = [\boldsymbol{\pi}^{(0)}, \ \boldsymbol{\pi}^{(1)}, \ \boldsymbol{\pi}^{(*)}]$, where $\boldsymbol{\pi}^{(*)} = \sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}$.

Proof. We first show that $[\boldsymbol{\pi}^{(0)}, \ \boldsymbol{\pi}^{(1)}, \ \boldsymbol{\pi}^{(*)}]$ is a solution of Eq.(5.2) by verifying that it satisfies four matrix equations corresponding to the four sets of columns we used to define $\mathbf{X}$.

(i) The first equation is the normalization constraint:

$$\boldsymbol{\pi}^{(0)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(1)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(*)}\cdot\mathbf{1}^T = 1. \tag{5.4}$$

(ii) The second set of m equations is the first line in Eq. (5.1):

$$\boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{L}} + \boldsymbol{\pi}^{(1)}\cdot\widehat{\mathbf{B}} = \mathbf{0}. \tag{5.5}$$

(iii) The third set of n equations is derived beginning from the second line in Eq.(5.1):

$$\boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{F}}^{(1)} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{B} = \mathbf{0}.$$

Further we rewrite $\boldsymbol{\pi}^{(2)}$, such that it is expressed in terms of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(*)}$ only, because our solution does not compute explicitly $\boldsymbol{\pi}^{(2)}$. By substituting $\pi^{(2)}$ in the above equation we obtain:

$$\pi^{(0)} \cdot \widehat{\mathbf{F}}^{(1)} + \pi^{(1)} \cdot \mathbf{L} + \pi^{(*)} \cdot \mathbf{B} - \sum_{i=3}^{\infty} \pi^{(i)} \cdot \mathbf{B} = \mathbf{0}. \tag{5.6}$$

To compute the sum $\sum_{i=3}^{\infty} \pi^{(i)}$, we use Ramaswami's recursive formula, i.e., Eq.(3.26), and obtain:

$$\begin{aligned}
\pi^{(3)} &= -(\pi^{(0)} \cdot \widehat{\mathbf{S}}^{(3)} + \pi^{(1)} \cdot \mathbf{S}^{(2)} + \pi^{(2)} \cdot \mathbf{S}^{(1)}) \quad \cdot (\mathbf{S}^{(0)})^{-1} \\
\pi^{(4)} &= -(\pi^{(0)} \cdot \widehat{\mathbf{S}}^{(4)} + \pi^{(1)} \cdot \mathbf{S}^{(3)} + \pi^{(2)} \cdot \mathbf{S}^{(2)} + \pi^{(3)} \cdot \mathbf{S}^{(1)}) \quad \cdot (\mathbf{S}^{(0)})^{-1} \\
\pi^{(5)} &= -(\pi^{(0)} \cdot \widehat{\mathbf{S}}^{(5)} + \pi^{(1)} \cdot \mathbf{S}^{(4)} + \pi^{(2)} \cdot \mathbf{S}^{(3)} + \pi^{(3)} \cdot \mathbf{S}^{(2)} + \pi^{(4)} \cdot \mathbf{S}^{(1)}) \cdot (\mathbf{S}^{(0)})^{-1} \cdot \\
&\vdots
\end{aligned} \tag{5.7}$$

where the matrices $\widehat{\mathbf{S}}^{(i)}$, for $i \geq 3$, and $\mathbf{S}^{(j)}$, for $j \geq 1$ are determined using the definitions in Eq.(3.27).

From the definition of matrix $\mathbf{G}$ in Eq.(3.25), it follows that

$$\mathbf{B} = -(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \mathbf{G}^{i}) \cdot \mathbf{G} = -\mathbf{S}^{(0)} \cdot \mathbf{G}.$$

After summing all equations in (5.7) and multiplying by $\mathbf{B}$, we obtain the sum $\sum_{i=3}^{\infty} \pi^{(i)} \cdot \mathbf{B}$:

$$\sum_{i=3}^{\infty} \pi^{(i)} \cdot \mathbf{B} = \left(\pi^{(0)} \cdot \sum_{i=3}^{\infty} \widehat{\mathbf{S}}^{(i)} + \pi^{(1)} \cdot \sum_{i=2}^{\infty} \mathbf{S}^{(i)} + \sum_{i=2}^{\infty} \pi^{(i)} \cdot \sum_{j=1}^{\infty} \mathbf{S}^{(j)}\right) \cdot (\mathbf{S}^{(0)})^{-1} \cdot \mathbf{S}^{(0)} \cdot \mathbf{G},$$

which further results in:

$$\sum_{i=3}^{\infty} \pi^{(i)} \cdot \mathbf{B} = \pi^{(0)} \cdot \sum_{i=3}^{\infty} \widehat{\mathbf{S}}^{(i)} \cdot \mathbf{G} + \pi^{(1)} \cdot \sum_{i=2}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G} + \sum_{i=2}^{\infty} \pi^{(i)} \cdot \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G}. \tag{5.8}$$

Substituting Eq.(5.8) in Eq.(5.6) we obtain the third set of equations as a function of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(*)}$ only:

$$\boldsymbol{\pi}^{(0)} \cdot \left(\widehat{\mathbf{F}}^{(1)} - \sum_{i=3}^{\infty} \widehat{\mathbf{S}}^{(i)} \cdot \mathbf{G} \right) + \boldsymbol{\pi}^{(1)} \cdot \left(\mathbf{L} - \sum_{i=2}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G} \right) + \boldsymbol{\pi}^{(*)} \cdot \left(\mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G} \right) = \mathbf{0}. \tag{5.9}$$

(iv) Another set of n equations is obtained by summing all the remaining lines in Eq.(5.1):

$$\boldsymbol{\pi}^{(0)} \cdot \sum_{i=2}^{\infty} \widehat{\mathbf{F}}^{(i)} + \boldsymbol{\pi}^{(1)} \cdot \sum_{i=1}^{\infty} \mathbf{F}^{(i)} + \sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \cdot \left(\mathbf{L} + \sum_{i=j}^{\infty} \mathbf{F}^{(j)} \right) + \sum_{i=3}^{\infty} \boldsymbol{\pi}^{(i)} \cdot \mathbf{B} = \mathbf{0}.$$

Since $\sum_{i=3}^{\infty} \boldsymbol{\pi}^{(i)} \cdot \mathbf{B}$ can be expressed as a function of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$ only, the above equation can be rewritten as:

$$\begin{aligned} \boldsymbol{\pi}^{(0)} \cdot \left(\sum_{i=2}^{\infty} \widehat{\mathbf{F}}^{(i)} + \sum_{i=3}^{\infty} \widehat{\mathbf{S}}^{(i)} \cdot \mathbf{G} \right) + \boldsymbol{\pi}^{(1)} \cdot \left(\sum_{i=1}^{\infty} \mathbf{F}^{(i)} + \sum_{i=2}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G} \right) & \\ + \boldsymbol{\pi}^{(*)} \cdot \left(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} + \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \cdot \mathbf{G} \right) &= \mathbf{0}. \end{aligned} \tag{5.10}$$

In steps (i) through (iv), we showed that the vector $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$ satisfies Eqs. (5.4), (5.5), (5.9), and (5.10), hence it is a solution of (5.2). Now we have to show that this solution is unique. For this, it is enough to prove that the rank of $\mathbf{X}$ is $m + 2n$ by showing that its $m + 2n$ rows are linearly independent.

Since the process with the infinitesimal generator $\mathbf{Q}_{M/G/1}$ is ergodic, we know that the vector $\mathbf{1}^T$ and the set of vectors corresponding to all the columns of $\mathbf{Q}_{M/G/1}$ except one, any of them, are linearly independent. We also note that by multiplying a block column of the infinitesimal generator $\mathbf{Q}_{M/G/1}$ with a matrix, we get a block column which is a linear combination of the columns of the selected block column. In our proof we use multiplication of the block columns with the powers of matrix $\mathbf{G}$.

$\mathbf{V}^{(0)}$	$\mathbf{V}^{(1)}$	$\mathbf{V}^{(2)}$	$\mathbf{V}^{(3)}$	$\cdots$
$\widehat{\mathbf{L}}$	$\widehat{\mathbf{F}}^{(1)}$	$\widehat{\mathbf{F}}^{(2)}$	$\widehat{\mathbf{F}}^{(3)}$	$\cdots$
$\widehat{\mathbf{B}}$	$\mathbf{L}$	$\mathbf{F}^{(1)}$	$\mathbf{F}^{(2)}$	$\cdots$
$\mathbf{0}$	$\mathbf{B}$	$\mathbf{L}$	$\mathbf{F}^{(1)}$	$\cdots$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{B}$	$\mathbf{L}$	$\cdots$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{B}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\mathbf{U}$
$\sum_{i=2}^{\infty}\widehat{\mathbf{F}}^{(i)}$
$\sum_{i=1}^{\infty}\mathbf{F}^{(i)}$
$\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}$
$\mathbf{B}+\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}$
$\mathbf{B}+\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}$
$\vdots$

$\mathbf{W}^{(1)}$	$\mathbf{W}^{(2)}$	$\mathbf{W}^{(3)}$	$\cdots$
$\widetilde{\mathbf{S}}^{(3)}\cdot\mathbf{G}$	$\widetilde{\mathbf{S}}^{(4)}\cdot\mathbf{G}$	$\widetilde{\mathbf{S}}^{(5)}\cdot\mathbf{G}$	
$\mathbf{S}^{(2)}\cdot\mathbf{G}$	$\mathbf{S}^{(3)}\cdot\mathbf{G}$	$\mathbf{S}^{(4)}\cdot\mathbf{G}$	
$\mathbf{S}^{(1)}\cdot\mathbf{G}$	$\mathbf{S}^{(2)}\cdot\mathbf{G}$	$\mathbf{S}^{(3)}\cdot\mathbf{G}$	
$-\mathbf{B}$	$\mathbf{S}^{(1)}\cdot\mathbf{G}$	$\mathbf{S}^{(2)}\cdot\mathbf{G}$	
$\mathbf{0}$	$-\mathbf{B}$	$\mathbf{S}^{(1)}\cdot\mathbf{G}$	
$\vdots$	$\vdots$	$\vdots$	

$\mathbf{Y}$
$\widehat{\mathbf{F}}^{(1)}-\sum_{i=3}^{\infty}\widehat{\mathbf{S}}^{(i)}\cdot\mathbf{G}$
$\mathbf{L}-\sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{B}-\sum_{i=1}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{B}-\sum_{i=1}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{B}-\sum_{i=1}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\vdots$

$\mathbf{Z}$
$\sum_{i=2}^{\infty}\widehat{\mathbf{F}}^{(i)}+\sum_{i=3}^{\infty}\widehat{\mathbf{S}}^{(i)}\cdot\mathbf{G}$
$\sum_{i=1}^{\infty}\mathbf{F}^{(i)}+\sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}+\sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}+\sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{F}^{(i)}+\sum_{i=2}^{\infty}\mathbf{S}^{(i)}\cdot\mathbf{G}^{i}$
$\vdots$

Figure 5.2: The blocks of column vectors used to prove linear independence.

In the following, we define all block columns that we use in our proof and, for the benefit of the reader, we show them in Figure 5.2. We label the original block columns of $\mathbf{Q}_{M/G/1}$ as $\mathbf{V}^{(i)}$ for $i \geq 0$. We define block column $\mathbf{U}$ as the sum of all $\mathbf{V}^{(i)}$ for $i \geq 2$:

$$\mathbf{U} = \sum_{i=2}^{\infty} \mathbf{V}^{(i)}.$$

We obtain blocks $\mathbf{W}^{(i)}$ for $i \geq 1$ by multiplying the block columns $\mathbf{V}^{(j)}$ for

$j \geq i+2$ with the i^{th} power of matrix $\mathbf{G}$ and summing them all together

$$\mathbf{W}^{(i)} = \sum_{j=3}^{\infty} \mathbf{V}^{(j)} \cdot \mathbf{G}^{i}, \quad i \geq 1.$$

Blocks $\mathbf{W}^{(i)}$ for $i \geq 1$ are further used to define

$$\mathbf{Y} = \mathbf{V}^{(1)} - \sum_{i=1}^{\infty} \mathbf{W}^{(i)}$$

and

$$\mathbf{Z} = \mathbf{U} + \sum_{i=1}^{\infty} \mathbf{W}^{(i)}.$$

In the matrix $\mathbf{X}$ defined in Eq.(5.3), we make use of the three upper blocks of $\mathbf{V}^{(0)}$, $\mathbf{Y}$, and $\mathbf{Z}$. We argue that the rank of the matrix $[\mathbf{V}^{(0)}|\mathbf{Y}|\mathbf{Z}]$ is $m+2n-1$ because we obtained $\mathbf{Y}$, and $\mathbf{Z}$ respectively as linear combination of blocks $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$ with the blocks $\mathbf{W}^{(i)}$ for $i \geq 1$, and none of the columns used to generate $\mathbf{W}^{(i)}$ for $i \geq 1$ is from either $\mathbf{V}^{(1)}$ or $\mathbf{V}^{(2)}$. Recall that $\mathbf{Q}_{M/G/1}$ is an infinitesimal generator, therefore the defect is one. Therefore, the rank of $[\mathbf{V}^{(0)}|\mathbf{Y}|\mathbf{Z}]$ is exactly $m+2n-1$. Substituting one (any) of these columns with a column of 1*s*, we obtain the rank of $m+2n$. □

5.1.1 Computing measures of interest for M/G/1-type processes

We now consider the problem of obtaining stationary measures of interest once $\pi^{(0)}$, $\pi^{(1)}$, and $\pi^{(*)}$ have been computed. We consider measures that can be expressed as the expected reward rate:

$$r = \sum_{j=0}^{\infty} \sum_{i \in S^{(j)}} \rho_i^{(j)} \pi_i^{(j)},$$

where $\rho_i^{(j)}$ is the *reward rate* of state $s_i^{(j)}$. For example, if we wanted to compute the expected queue length in steady state for a model, where $S^{(j)}$ contains the system states with j customers in the queue, we would let $\rho_i^{(j)} = j$, while, to compute the second moment of the queue length, we would let $\rho_i^{(j)} = j^2$.

Since our solution approach computes $\pi^{(0)}$, $\pi^{(1)}$, and $\sum_{j=2}^{\infty} \pi^{(j)}$, we rewrite r as

$$r = \pi^{(0)} \rho^{(0)T} + \pi^{(1)} \rho^{(1)T} + \sum_{j=2}^{\infty} \pi^{(j)} \rho^{(j)T},$$

where $\rho^{(0)} = [\rho_1^{(0)}, \ldots, \rho_m^{(0)}]$ and $\rho^{(j)} = [\rho_1^{(j)}, \ldots, \rho_n^{(j)}]$, for $j \geq 1$. Then, we must show how to compute the above summation without explicitly using the values of $\pi^{(j)}$ for $j \geq 2$. We can do so if the reward rate of state $s_i^{(j)}$, for $j \geq 2$ and $i = 1, \ldots, n$, is a polynomial of degree k in j with arbitrary coefficients $\mathbf{a}_i^{[0]}, \mathbf{a}_i^{[1]}, \ldots, \mathbf{a}_i^{[k]}$:

$$\forall j \geq 2, \ \forall i \in \{1, 2, \ldots, n\}, \qquad \rho_i^{(j)} = \mathbf{a}_i^{[0]} + \mathbf{a}_i^{[1]} j + \cdots + \mathbf{a}_i^{[k]} j^k. \tag{5.11}$$

The definition of $\rho_i^{(j)}$ illustrates that the set of measures of interest that we can compute includes any moment of the probability vector $\boldsymbol{\pi}$ as long as the reward rate of the i^{th} state in each set $\mathcal{S}^{(j)}$ has the same polynomial coefficients for all $j \geq 2$.

We compute $\sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \boldsymbol{\rho}^{(j)T}$ as follows

$$\begin{aligned}
\sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \boldsymbol{\rho}^{(j)T} &= \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \left(\mathbf{a}^{[0]} + \mathbf{a}^{[1]} j + \cdots + \mathbf{a}^{[k]} j^k\right)^T \\
&= \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \mathbf{a}^{[0]T} + \sum_{j=2}^{\infty} j \boldsymbol{\pi}^{(j)} \mathbf{a}^{[1]T} + \cdots + \sum_{j=2}^{\infty} j^k \boldsymbol{\pi}^{(j)} \mathbf{a}^{[k]T} \\
&= \mathbf{r}^{[0]} \mathbf{a}^{[0]T} + \mathbf{r}^{[1]} \mathbf{a}^{[1]T} + \cdots + \mathbf{r}^{[k]} \mathbf{a}^{[k]T},
\end{aligned}$$

and the problem is reduced to the computation of $\mathbf{r}^{[l]} = \sum_{j=2}^{\infty} j^l \boldsymbol{\pi}^{(j)}$, for $l = 0, \ldots, k$.

We show how $\mathbf{r}^{[k]}$, $k > 0$, can be computed recursively, starting from $\mathbf{r}^{[0]}$, which is simply $\boldsymbol{\pi}^{(*)}$. Multiplying the equations in (5.1) from the second line on by the appropriate factor j^k results in

$$\left\{ \begin{array}{lllllllll}
2^k \boldsymbol{\pi}^{(0)} & \widehat{\mathbf{F}}^{(1)} & + & 2^k \boldsymbol{\pi}^{(1)} & \mathbf{L}^{(1)} & + & 2^k \boldsymbol{\pi}^{(2)} & \mathbf{B} & & & = \mathbf{0} \\
3^k \boldsymbol{\pi}^{(0)} & \widehat{\mathbf{F}}^{(2)} & + & 3^k \boldsymbol{\pi}^{(1)} & \mathbf{F}^{(1)} & + & 3^k \boldsymbol{\pi}^{(2)} & \mathbf{L} & + & 3^k \boldsymbol{\pi}^{(3)} \;\; \mathbf{B} & = \mathbf{0} \, . \\
 & & & & \vdots & & & & & &
\end{array} \right.$$

Summing these equations by parts we obtain

$$\underbrace{\boldsymbol{\pi}^{(0)} \sum_{j=1}^{\infty} (j+1)^k \widehat{\mathbf{F}}^{(j)}}_{\stackrel{\text{def}}{=} \hat{\mathbf{f}}} + \underbrace{\boldsymbol{\pi}^{(1)} \left(2^k \mathbf{L} + \sum_{j=1}^{\infty} (j+2)^k \mathbf{F}^{(j)} \right)}_{\stackrel{\text{def}}{=} \mathbf{f}} +$$

$$\sum_{h=2}^{\infty} \boldsymbol{\pi}^{(h)} \left(\sum_{j=1}^{\infty} (j+h+1)^k \mathbf{F}^{(j)} + (h+1)^k \mathbf{L} \right) + \underbrace{\sum_{h=2}^{\infty} \boldsymbol{\pi}^{(h)} h^k}_{= \mathbf{r}^{[k]}} \mathbf{B} = \mathbf{0},$$

which can then be rewritten as

$$\sum_{h=2}^{\infty} \boldsymbol{\pi}^{(h)} \left[\left(\sum_{j=1}^{\infty} \sum_{l=0}^{k} \binom{k}{l} (j+1)^l h^{k-l} \mathbf{F}^{(j)} \right) + \left(\sum_{l=0}^{k} \binom{k}{l} 1^l h^{k-l} \mathbf{L} \right) \right] + \mathbf{r}^{[k]} \mathbf{B} = -\hat{\mathbf{f}} - \mathbf{f}.$$

Exchanging the order of summations we obtain

$$\sum_{l=0}^{k} \binom{k}{l} \underbrace{\sum_{h=2}^{\infty} \boldsymbol{\pi}^{(h)} h^{k-l}}_{= \mathbf{r}^{[k-l]}} \left(\mathbf{L} + \sum_{j=1}^{\infty} (j+1)^l \mathbf{F}^{(j)} \right) + \mathbf{r}^{[k]} \mathbf{B} = -\hat{\mathbf{f}} - \mathbf{f}.$$

Finally, isolating the case $l = 0$ in the outermost summation we obtain

$$\mathbf{r}^{[k]} \left(\mathbf{B} + \mathbf{L} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)} \right) = -\hat{\mathbf{f}} - \mathbf{f} - \sum_{l=1}^{k} \binom{k}{l} \mathbf{r}^{[k-l]} \left(\mathbf{L} + \sum_{j=1}^{\infty} (j+1)^l \mathbf{F}^{(j)} \right).$$

which is a linear system of the form $\mathbf{r}^{[k]}(\mathbf{B} + \mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(j)}) = \mathbf{b}^{[k]}$, where the right-hand side $\mathbf{b}^{[k]}$ is an expression that can be effectively computed from $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and the vectors $\mathbf{r}^{[0]}$ through $\mathbf{r}^{[k-1]}$. However, the rank of $(\mathbf{B}+\mathbf{L}+\sum_{i=1}^{\infty} \mathbf{F}^{(j)})$ is $n-1$. This is true because $(\mathbf{B}+\mathbf{L}+\sum_{i=1}^{\infty} \mathbf{F}^{(j)})$ is an infinitesimal generator with rank $n-1$, so the above system is under-determined. We drop any of the columns of $\mathbf{B}+\mathbf{L}+\sum_{i=1}^{\infty} \mathbf{F}^{(j)}$, resulting in

$$\mathbf{r}^{[k]}(\mathbf{B} + \mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(j)})^{\circ} = (\mathbf{b}^{[k]})^{\circ}. \qquad (5.12)$$

and obtain one additional equation for $\mathbf{r}^{[k]}$ by using the flow balance equations between $\cup_{l=0}^{j} \mathcal{S}^{(l)}$ and $\cup_{l=j+1}^{\infty} \mathcal{S}^{(l)}$ for each $j \geq 1$ and multiplying them by the appropriate factor

j^k,

$$\begin{cases} 2^k\boldsymbol{\pi}^{(0)} \sum_{l=2}^{\infty}\widehat{\mathbf{F}}^{(l)}\mathbf{1}^T + 2^k\boldsymbol{\pi}^{(1)} \sum_{l=1}^{\infty}\mathbf{F}^{(l)}\mathbf{1}^T & = 2^k\boldsymbol{\pi}^{(2)}\ \mathbf{B}\mathbf{1}^T \\ 3^k\boldsymbol{\pi}^{(0)} \sum_{l=3}^{\infty}\widehat{\mathbf{F}}^{(l)}\mathbf{1}^T + 3^k\boldsymbol{\pi}^{(1)} \sum_{l=2}^{\infty}\mathbf{F}^{(l)}\mathbf{1}^T + 3^k\boldsymbol{\pi}^{(2)} \sum_{l=1}^{\infty}\mathbf{F}^{(l)}\mathbf{1}^T & = 3^k\boldsymbol{\pi}^{(3)}\ \mathbf{B}\mathbf{1}^T \ \cdot \\ \vdots & \end{cases} \tag{5.13}$$

We introduce the following notation

$$\widehat{\mathbf{F}}_{[k,j]} = \sum_{l=j}^{\infty} l^k \cdot \widehat{\mathbf{F}}^{(l)}, \qquad \mathbf{F}_{[k,j]} = \sum_{l=j}^{\infty} l^k \cdot \mathbf{F}^{(l)}, \qquad j \geq 1. \tag{5.14}$$

We then sum all lines in Eq.(5.13) and obtain:

$$\boldsymbol{\pi}^{(0)} \sum_{j=1}^{\infty}(j+1)^k\widehat{\mathbf{F}}_{[0,j+1]}\mathbf{1}^T + \boldsymbol{\pi}^{(1)}\sum_{j=0}^{\infty}(j+2)^k\mathbf{F}_{[0,j+1]}\mathbf{1}^T +$$

$$\sum_{l=2}^{\infty}\boldsymbol{\pi}^{(l)}\sum_{j=1}^{\infty}(j+l)^k\mathbf{F}_{[0,j]}\mathbf{1}^T = \sum_{j=2}^{\infty} j^k\boldsymbol{\pi}^{(j)}\mathbf{B}\mathbf{1}^T.$$

which, with steps analogous to those just performed to obtain Eq.(5.12), can be written as

$$\mathbf{r}^{[k]}(\mathbf{F}_{[1,1]} - \mathbf{B})\mathbf{1}^T = c^{[k]}, \tag{5.15}$$

where $c^{[k]}$ is, again, an expression containing $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and the vectors $\mathbf{r}^{[0]}$ through $\mathbf{r}^{[k-1]}$.

Note that the $n \times n$ matrix

$$[(\mathbf{B} + \mathbf{L} + \mathbf{F}_{[0,1]})^{\circ}|(\mathbf{F}_{[1,1]} - \mathbf{B})\mathbf{1}^T] \tag{5.16}$$

has full rank. This is true because $(\mathbf{B}+\mathbf{L}+\mathbf{F}_{[0,1]})$ is an infinitesimal generator with rank $n-1$, thus has a unique stationary probability vector $\boldsymbol{\gamma}$ satisfying $\boldsymbol{\gamma}(\mathbf{B}+\mathbf{L}+\mathbf{F}_{[0,1]}) = \mathbf{0}$. However, this same vector must satisfy $\boldsymbol{\gamma}\mathbf{B}\mathbf{1}^T > \boldsymbol{\gamma}\mathbf{F}_{[1,1]}\mathbf{1}^T$ to ensure that the process has a positive drift toward $\mathcal{S}^{(0)}$, thus is ergodic, hence $\boldsymbol{\gamma}(\mathbf{F}_{[1,1]}-\mathbf{B})\mathbf{1}^T < 0$, which shows that $(\mathbf{F}_{[1,1]}-\mathbf{B})\mathbf{1}^T$ cannot be possibly obtained as linear combination of columns in $(\mathbf{B}+\mathbf{L}+\mathbf{F}_{[0,1]})$, therefore the $n \times n$ matrix defined in Eq.(5.16) has full rank.

Hence, we can compute $\mathbf{r}^{[k]}$ using Eqs. (5.12) and (5.15), i.e., solving a linear system in n unknowns (of course, we must do so first for $l = 1, \ldots, k-1$).

As an example, we consider $\mathbf{r}^{[1]}$, which is used to compute measures such as the first moment of the queue length. In this case,

$$\mathbf{b}^{[1]} = -\boldsymbol{\pi}^{(0)}\sum_{j=1}^{\infty}(j+1)\widehat{\mathbf{F}}^{(j)} - \boldsymbol{\pi}^{(1)}(2\mathbf{L}+\sum_{j=1}^{\infty}(j+2)\mathbf{F}^{(j)}) - \boldsymbol{\pi}^{(*)}(\mathbf{L}+\sum_{j=1}^{\infty}(j+1)\mathbf{F}^{(j)}),$$

$$c^{[1]} = -\boldsymbol{\pi}^{(0)}\sum_{j=2}^{\infty}j\widehat{\mathbf{F}}_{[0,j]}\mathbf{1}^T - \boldsymbol{\pi}^{(1)}\sum_{j=1}^{\infty}(j+1)\mathbf{F}_{[0,j]}\mathbf{1}^T - \boldsymbol{\pi}^{(*)}\sum_{j=1}^{\infty}j\mathbf{F}_{[0,j]}\mathbf{1}^T.$$

We conclude by observing that, when the sequences $\{\widehat{\mathbf{F}}^{(j)} : j \geq 1\}$ and $\{\mathbf{F}^{(j)} : j \geq 1\}$ do have a nicer relation, like a geometric one, the treatment in this section can be modified appropriately to simplify the different sums introduced here, and give closed form formulas.

In the general case, that was considered here, some measures might be infinite. For example, if the sequences are summable but decrease only like $1/j^h$ for some $h > 1$, then the moments of order $h-1$ or higher for the queue length do not exist (are infinite). From the practical point of view, we always store a finite set of matrices from the sequences $\{\widehat{\mathbf{F}}^{(j)} : j \geq 1\}$ and $\{\mathbf{F}^{(j)} : j \geq 1\}$, so the sums of type $\widehat{\mathbf{F}}_{[k,j]}$ and $\mathbf{F}_{[k,j]}$ for $j \geq 1, k \geq 0$ are always finite.

5.1.2 Time and storage complexity

In this section, we present a detailed comparison of our aggregate solution for M/G/1-type processes with the matrix-analytic method outlined in Subsection 3.4. The complexity analysis is within the accuracy of O-notation. In our analysis, $O^L(x)$ denotes the time complexity of solving a linear system described by x nonzero entries, and $\eta\{\mathbf{A}\}$ denotes the number of nonzero entries in matrix $\mathbf{A}$. In the general case, $\eta\{\widehat{\mathbf{F}}\}$ and $\eta\{\mathbf{F}\}$ should be taken to mean $\eta\{\cup_{j=1}^{p}\widehat{\mathbf{F}}^{(j)}\}$ and $\eta\{\cup_{j=1}^{p}\mathbf{F}^{(j)}\}$, respectively.

Since practically, we cannot store an infinite number of matrices, in our analysis, we store up to p matrices of type $\widehat{\mathbf{F}}^{(j)}$, and $\mathbf{F}^{(j)}$, $j \geq 1$, assuming that these matrices capture the behavior of the entire system. In addition for the matrix analytic method, to reach the necessary accuracy, we compute up to s block vectors $\boldsymbol{\pi}^{(i)}$ of the stationary probability vector $\boldsymbol{\pi}$.

We outline the required steps for each method and analyze the computation and storage complexity of each step up to the computation of the expected queue length of the process. In our analysis we do not include the cost to compute the matrix $\mathbf{G}$ since in both methodologies it is required to be computed. $\mathbf{G}$ should be computed with an efficient method like the cyclic-reduction algorithm [12].

The following summarizes the analysis of computation and storage requirements for ETAQA-M/G/1:

- Computation of the aggregate stationary probability vector $\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}$
  - $O(p \cdot (m \cdot \eta\{\widehat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\mathbf{F}, \mathbf{G}\}))$ to compute sums of the form $\widehat{\mathbf{S}}^{(i)}$ for $i \geq 1$, and $\mathbf{S}^{(i)}$ for $i \geq 0$, whose sparsity depends directly on the sparsity of $\mathbf{G}$, $\widehat{\mathbf{F}}^{(i)}$ and $\mathbf{F}^{(i)}$ for $i \geq 1$,
  - $O(p \cdot (\eta\{\widehat{\mathbf{F}}\} + \eta\{\mathbf{F}\}))$ to compute sums of the form $\sum_{j=1}^{\infty} \mathbf{F}^{(j)}$, and $\sum_{j=2}^{\infty} \widehat{\mathbf{F}}^{(j)}$.
  - $O^{L}(\eta\{\widehat{\mathbf{B}}, \widehat{\mathbf{L}}, \mathbf{L}, \widehat{\mathbf{F}}, \mathbf{F}, \mathbf{G}\})$ for the solution of the system of $m + 2n$ linear equations.
- Storage requirements for computation of $\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}$
  - $O(m \cdot n + n^2)$ to store the sums $\sum_{i=1}^{\infty} \widehat{\mathbf{S}}^{(i)}$ and $\sum_{i=1}^{\infty} \mathbf{S}^{(i)}$.
  - $m + 2n$ to store the probability vectors $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(*)}$.
- Computation of the expected queue length
  - $O(p \cdot (\eta\{\widehat{\mathbf{F}}\} + \eta\{\mathbf{F}\}))$ to compute sums of the form $\sum_{j=1}^{\infty} j^k \cdot \mathbf{F}^{(j)}$, and $\sum_{j=2}^{\infty} j^k \cdot \widehat{\mathbf{F}}^{(j)}$ where k is a constant,
  - $O^{L}(\eta\{\mathbf{F}, \mathbf{L}, \mathbf{B}\})$ for the solution of the sparse system of n linear equations.
- No additional storage requirements.

The following outlines the computational and storage complexity analysis for matrix-analytic solution for M/G/1-type processes presented in Section 3.7:

- Computation of boundary stationary probability vector $\boldsymbol{\pi}^{(0)}$
  - $O(p \cdot (m \cdot \eta\{\widehat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\mathbf{F}, \mathbf{G}\}))$ to compute the sums of the form $\widehat{\mathbf{S}}^{(i)}$ for $i \geq 1$, and $\mathbf{S}^{(i)}$ for $i \geq 0$,
  - $O(n^3 + m \cdot \eta\{\widehat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\widehat{\mathbf{B}}\})$ for the computation of the inverses of $\mathbf{S}^{(0)}$, and $\sum_{j=0}^{\infty} \mathbf{S}^{(j)}$ and additional full-matrix multiplications.
  - $O^L(m^2)$ for the solution of the system of m linear equations.
- Storage requirements for computation of $\boldsymbol{\pi}^{(0)}$
  - $O(p \cdot (m \cdot n + n^2))$ to store all sums of form $\widehat{\mathbf{S}}^{(i)}$ for $i \geq 1$, and $\mathbf{S}^{(i)}$ for $i \geq 0$,
  - $O(m^2)$ for storing the matrix of the system of linear equation.
  - m to store $\boldsymbol{\pi}^{(0)}$.
- Computation of the expected queue length
  - $O(pn^3 + sn^2 + p \log p)$ based on [58], since we assume that the FFT-based version of Ramaswami's recursive formula is used to compute the s vectors of the stationary probability vector,
  - $O(s \cdot n)$ to compute the queue length.
- Storage requirement: $s \cdot n$ to store vectors $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$.

Tables 5.1 and 5.2 summarize the discussion in this section.

Computation of $\pi^{(0)}$ (matrix-analytic) or $\pi^{(0)}$, $\pi^{(1)}$, $\pi^{(*)}$ (Etaqa-M/G/1)	
ETAQA-M/G/1	$O^L(\eta\{\hat{\mathbf{B}}, \hat{\mathbf{L}}, \mathbf{L}, \hat{\mathbf{F}}, \mathbf{F}, \mathbf{G}\}) + O(p \cdot (m \cdot \eta\{\hat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\mathbf{F}, \mathbf{G}\}))$
Matrix-analytic	$O^L(m^2) + O(p \cdot (m \cdot \eta\{\hat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\mathbf{F}, \mathbf{G}\}) + n^3 + m \cdot \eta\{\hat{\mathbf{F}}, \mathbf{G}\} + n \cdot \eta\{\hat{\mathbf{B}}\})$
First moment measures	
ETAQA-M/G/1	$O^L(\eta\{\mathbf{B}, \mathbf{L}, \mathbf{F}\}) + O(p \cdot \eta(\hat{\mathbf{F}}) + p \cdot \eta(\mathbf{F}))$
Matrix-analytic	$O(pn^3 + sn^2 + p \log p)$

Table 5.1: Computational complexity of the ETAQA-M/G/1 solution and the matrix-analytic method.

	Additional storage	**Storage of the probabilities**
Computation of $\pi^{(0)}$ (matrix-analytic) or $\pi^{(0)}$, $\pi^{(1)}$, $\pi^{(*)}$ (Etaqa-M/G/1)		
ETAQA-M/G/1	$O(m \cdot n + n^2)$	$m + 2n$
Matrix-analytic	$O(m^2 + p \cdot (m \cdot n + n^2))$	m
First moment measures		
ETAQA-M/G/1	none	none
Matrix-analytic	none	$s \cdot n$

Table 5.2: Storage complexity of the ETAQA-M/G/1 solution and the matrix-analytic method.

Concluding our analysis, we point out that the aggregate solution is a more efficient approach, both computation- and storage-wise. In comparison to the matrix-analytic solution, it entails only a few steps and is thus much easier to implement. Since we do not need to generate the whole stationary probability vector, in our complexity analysis the term s does not appear for ETAQA-M/G/1 which in comparison with the value of p or n is several times higher.

Furthermore, since the aggregate solution does not introduce any matrix inverse or matrix multiplication, the sparsity of the original process is preserved resulting in significant savings with respect to both computation and storage. We stress that the sparsity of $\mathbf{G}$ is key for preserving the sparsity of the original process in both methods. There are special cases where $\mathbf{G}$ is very sparse (e.g., $\mathbf{G}$ is a single column matrix if $\mathbf{B}$ is a single column matrix). In these cases, the sums of the form $\widehat{\mathbf{S}}^{(i)}$ for $i \geq 1$, and $\mathbf{S}^{(i)}$ for $i \geq 0$ almost preserve the sparsity of the original process and reduce the computation and storage cost.

5.2 Etaqa-GI/M/1

In this section, we apply the same aggregation technique, introduced in Section 5.1, for the exact solution of GI/M/1-type processes. Recall that in Eq.(2.21) we defined the block partitioned infinitesimal generator of GI/M/1-type processes as:

$$\mathbf{Q}_{GI/M/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The respective block partitioning of the stationary probability vector $\boldsymbol{\pi}$ allows us to rewrite the matrix equality $\boldsymbol{\pi} \cdot \mathbf{Q}_{GI/M/1} = \mathbf{0}$ as:

$$\left\{ \begin{array}{rcrcrcl} & & \boldsymbol{\pi}^{(0)}\widehat{\mathbf{L}} & + & \sum_{i=1}^{\infty} \boldsymbol{\pi}^{(i)}\widehat{\mathbf{B}}^{(i)} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(0)}\widehat{\mathbf{F}} & + & \boldsymbol{\pi}^{(1)}\mathbf{L} & + & \sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)}\mathbf{B}^{(i-1)} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(1)}\mathbf{F} & + & \boldsymbol{\pi}^{(2)}\mathbf{L} & + & \sum_{i=3}^{\infty} \boldsymbol{\pi}^{(i)}\mathbf{B}^{(i-2)} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(2)}\mathbf{F} & + & \boldsymbol{\pi}^{(3)}\mathbf{L} & + & \sum_{i=4}^{\infty} \boldsymbol{\pi}^{(i)}\mathbf{B}^{(i-3)} & = & \mathbf{0} \\ & & & & \vdots & & \end{array} \right. . \tag{5.17}$$

Assuming that matrix $\mathbf{R}$ is available, we apply the same steps as for the case of M/G/1-type processes and formulate the following theorem:

Theorem Given an ergodic CTMC with infinitesimal generator $\mathbf{Q}_{GI/M/1}$ having the structure shown in Eq.(2.21), with stationary probability vector $\boldsymbol{\pi} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, ...]$, the system of linear equations

$$\mathbf{x} \cdot \mathbf{X} = [1, \mathbf{0}], \tag{5.18}$$

where $\mathbf{X} \in \mathbb{R}^{(m+2n)\times(m+2n)}$ is defined as follows

$$\mathbf{X} = \left[\begin{array}{c|c|c|c} \mathbf{1}^T & \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0}^\circ \\ \mathbf{1}^T & \widehat{\mathbf{B}}^{(1)} & \mathbf{L} & \mathbf{F}^\circ \\ \mathbf{1}^T & (\mathbf{I}-\mathbf{R})\sum_{i=2}^{\infty}\mathbf{R}^{i-2}\cdot\widehat{\mathbf{B}}^{(i)} & (\mathbf{I}-\mathbf{R})\sum_{i=1}^{\infty}\mathbf{R}^{i-1}\cdot\mathbf{B}^{(i)} & (\mathbf{F}+\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{R}^{i}\cdot\mathbf{B}^{(i)})^\circ \end{array}\right]. \tag{5.19}$$

admits a unique solution $\mathbf{x} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$, where $\boldsymbol{\pi}^{(*)} = \sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}$.

Proof. The steps to derive Eq.(5.19) are outlined as follows.

(i) The first equation is the normalization constraint:

$$\boldsymbol{\pi}^{(0)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(1)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(*)}\cdot\mathbf{1}^T = 1. \tag{5.20}$$

(ii) From the first line in Eq.(5.17) we have:

$$\boldsymbol{\pi}^{(0)}\cdot\widehat{\mathbf{L}} + \boldsymbol{\pi}^{(1)}\cdot\widehat{\mathbf{B}}^{(1)} + \sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}\cdot\widehat{\mathbf{B}}^{(i)} = \mathbf{0}.$$

The sum $\sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}\cdot\widehat{\mathbf{B}}^{(i)}$ can be expressed as:

$$\sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}\cdot\widehat{\mathbf{B}}^{(i)} = \sum_{i=2}^{\infty}(\sum_{j=i}^{\infty}\boldsymbol{\pi}^{(j)} - \sum_{j=i+1}^{\infty}\boldsymbol{\pi}^{(j)})\cdot\widehat{\mathbf{B}}^{(i)}$$

and after simple derivations that exploit the geometric relation of the stationary probability vectors $\boldsymbol{\pi}^{(j)}$, for $j \geq 2$, we obtain m equations:

$$\boldsymbol{\pi}^{(0)} \cdot \widehat{\mathbf{L}} + \boldsymbol{\pi}^{(1)} \cdot \widehat{\mathbf{B}}^{(1)} + \boldsymbol{\pi}^{(*)}(\mathbf{I} - \mathbf{R}) \sum_{i=2}^{\infty} \mathbf{R}^{i-2} \cdot \widehat{\mathbf{B}}^{(i)}.$$

(iii) From the second line of Eq.(5.17) and using similar derivations as in step (ii) we get the third set of n equations:

$$\boldsymbol{\pi}^{(0)} \cdot \widehat{\mathbf{F}} + \boldsymbol{\pi}^{(1)} \cdot \mathbf{L} + \boldsymbol{\pi}^{(*)}(\mathbf{I} - \mathbf{R}) \sum_{i=1}^{\infty} \mathbf{R}^{i-1} \cdot \mathbf{B}^{(i)}.$$

(iv) Another set of n equations is obtained by summing all the remaining lines in Eq.(5.17):

$$\boldsymbol{\pi}^{(1)} \cdot \mathbf{F} + \boldsymbol{\pi}^{(*)} \cdot (\mathbf{L} + \mathbf{F}) + \sum_{i=3}^{\infty} \sum_{j=i}^{\infty} \boldsymbol{\pi}^{(j)} \mathbf{B}^{(i-2)}$$

and by expressing the sum $\sum_{i=3}^{\infty} \sum_{j=i}^{\infty} \boldsymbol{\pi}^{(j)} \mathbf{B}^{(i-2)}$ as a function of $\boldsymbol{\pi}^{(*)}$, we obtain an additional set of n equations:

$$\boldsymbol{\pi}^{(1)} \cdot \mathbf{F} + \boldsymbol{\pi}^{(*)} \left(\mathbf{L} + \mathbf{F} + \sum_{i=1}^{\infty} \mathbf{R}^{i} \cdot \mathbf{B}^{(i)} \right).$$

The matrix $\mathbf{X}$ has full rank. This follows from the fact that the infinitesimal generator $\mathbf{Q}_{GI/M/1}$ has a defect of one. We obtained the second and the third block columns in $\mathbf{X}$ by keeping their respective first two upper blocks in the first block column of $\mathbf{Q}_{GI/M/1}$ and substituting the remaining lower blocks with one block that results as a linear combination of the remaining lower blocks within the same block column of $\mathbf{Q}_{GI/M/1}$. We obtained the fourth block column in $\mathbf{X}$ by keeping the first two upper blocks from the third block column of $\mathbf{Q}_{GI/M/1}$ and substituting the rest with one block that results as a linear combination of the remaining lower blocks of the third block column in $\mathbf{Q}_{GI/M/1}$ plus all remaining blocks in $\mathbf{Q}_{GI/M/1}$ (i.e., from the fourth block column of $\mathbf{Q}_{GI/M/1}$ onward). Substituting one (any) of these columns with a column of 1*s*, we obtain the rank of $m+2n$.

□

5.2.1 Computing measures of interest for GI/M/1-type processes

Similarly to the M/G/1 case, our methodology allows the computation of the reward rate of state $s_i^{(j)}$, for $j \geq 2$ and $i = 1, \ldots, n$, if it is a polynomial of degree k in j with arbitrary coefficients $\mathbf{a}_i^{[0]}, \mathbf{a}_i^{[1]}, \ldots, \mathbf{a}_i^{[k]}$:

$$\forall j \geq 2, \ \forall i \in \{1, 2, \ldots, n\}, \qquad \rho_i^{(j)} = \mathbf{a}_i^{[0]} + \mathbf{a}_i^{[1]} j + \cdots + \mathbf{a}_i^{[k]} j^k.$$

We follow the exact same steps as those presented in Section 5.1.1. $\mathbf{r}^{[k]}$ is obtained by solving the system of linear equations

$$\mathbf{r}^{[k]}[\mathbf{F}+\mathbf{L}+\sum_{i=1}^{\infty}\mathbf{R}^i\mathbf{B}^{(i)})^\circ \mid ((\mathbf{I}-\mathbf{R})\sum_{i=2}^{\infty}\mathbf{R}^{i-2}\sum_{j=1}^{\infty}\mathbf{R}^i\cdot\widehat{\mathbf{B}}^{(j+2)}+\sum_{i=1}^{\infty}\sum_{j=i}^{\infty}\mathbf{R}^{j-1}\mathbf{B}^{(j)}-\mathbf{F})\cdot\mathbf{1}^T] = [(\mathbf{b}^{[k]})^\circ \mid c^{[k]}], \tag{5.21}$$

where

$$\mathbf{b}^{[k]} = -(\boldsymbol{\pi}^{(1)}\cdot 2^k\mathbf{F}+\sum_{l=0}^{k-1}\binom{k}{l}\mathbf{r}^{[l]}\cdot\mathbf{F})$$

and

$$c^{[k]} = -\sum_{l=0}^{k-1}\binom{k}{l}\mathbf{r}^{[l]}\cdot\left(\sum_{i=2}^{\infty}((i-2)^{k-l}\mathbf{I}-(i-1)^{k-l}\mathbf{R})\sum_{j=1}^{\infty}\mathbf{R}^i\cdot\widehat{\mathbf{B}}^{(j+2)}+\right.$$
$$\left.\sum_{i=1}^{\infty}\sum_{j=i+2}^{\infty}i^{k-l}\cdot\mathbf{R}^{j-1}\mathbf{B}^{(j)}\right).$$

The $n\times n$ matrix used in Eq.(5.21) has full rank. The proof follows the same steps as those used for the proof of Theorem (5.2).

5.2.2 Time and storage complexity

In this section, we present a detailed comparison of our ETAQA-GI/M/1 solution for GI/M/1-type processes with the matrix geometric solution outlined in Subsection 3.1. The complexity analysis is within the accuracy of O-notation. We assume that up to p of the $\widehat{\mathbf{B}}^{(j)}$, and $\mathbf{B}^{(j)}$, $j\geq 1$ matrices are stored. The notation in this section follows the one defined in section 5.1.2.

We outline the required steps for each method and analyze the computation and storage complexity of each step up to the computation of the expected queue length. Since both methods require **R**, we do not include this cost in our analysis and assume that is computed using an efficient method.

In the following, we summarize the complexity analysis of ETAQA-GI/M/1 solution method:

- Computation of the aggregate stationary probability vectors $\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}$
  - $O(p \cdot (m \cdot \eta\{\widehat{\mathbf{B}}, \mathbf{R}\} + n \cdot \eta\{\mathbf{B}, \mathbf{R}\}))$ for computation of sums of the form $\sum_{i=2}^{\infty} \mathbf{R}^{i-j}\widehat{\mathbf{B}}^{(i)}$ for $j = 1, 2$, and $\sum_{i=1}^{\infty} \mathbf{R}^{i-j}\mathbf{B}^{(i)}$ for $j = 0, 1$.
  - $O^L(\eta\{\widehat{\mathbf{L}}, \widehat{\mathbf{F}}, \widehat{\mathbf{B}}, \mathbf{L}, \mathbf{B}, \mathbf{R}\})$ for the solution of a system of $m + 2n$ linear equations.
- Storage requirements for computation of $\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(*)}$
  - $O(m \cdot n + n^2)$ to store sums of form $\sum_{i=2}^{\infty} \mathbf{R}^{i-j}\widehat{\mathbf{B}}^{(i)}$ for $j = 1, 2$ and $\sum_{i=1}^{\infty} \mathbf{R}^{i-j}\mathbf{B}^{(i)}$ for $j = 0, 1$,
  - n^2 to store matrix **R**,
  - $m + 2n$ to store $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(*)}$.
- Computation of the queue length
  - $O^L(\eta\{\mathbf{F}, \mathbf{L}, \mathbf{B}, \mathbf{R}\})$ to solve a system of n linear equations,

- $O(p^2(m \cdot \eta\{\widehat{\mathbf{B}}, \mathbf{R}\} + n \cdot \eta\{\mathbf{B}, \mathbf{R}\})$ for the sums required to construct the matrices of the system of linear equations.

- Storage requirements for computation of queue length
  - No additional requirements.

In the following, we summarize the complexity of matrix-geometric solution outlined in Section 3.1:

- Computation of the boundary stationary probability vectors $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$
  - $O(p\cdot(m\cdot\eta\{\widehat{\mathbf{B}}, \mathbf{R}\}+n\cdot\eta\{\mathbf{B}, \mathbf{R}\}))$ to compute sums of the form $\sum_{i=2}^{\infty} \mathbf{R}^{i-j}\widehat{\mathbf{B}}^{(i)}$ for $j = 1, 2$ and $\sum_{i=1}^{\infty} \mathbf{R}^{i-j}\mathbf{B}^{(i)}$ for $j = 0, 1$,
  - $O(n^3)$ to compute of $(\mathbf{I} - \mathbf{R})^{-1}$,
  - $O^L(\eta\{\widehat{\mathbf{L}}, \widehat{\mathbf{F}}, \widehat{\mathbf{B}}, \mathbf{L}, \mathbf{F}, \mathbf{B}, \mathbf{R}\})$ for the solution of a system of $m + n$ linear equations.

- Storage requirements for computation of $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$
  - $O(m \cdot n + n^2)$ to store sums of the form $\sum_{i=2}^{\infty} \mathbf{R}^{i-j}\widehat{\mathbf{B}}^{(i)}$ for $j = 1, 2$ and $\sum_{i=1}^{\infty} \mathbf{R}^{i-j}\mathbf{B}^{(i)}$ for $j = 0, 1$,
  - $O(n^2)$ to store $\mathbf{R}$ and $(\mathbf{I} - \mathbf{R})^{-1}$,
  - $m + n$ to store $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$.

- Computation of queue length

- $O(n^2)$ to compute the closed-form formula for queue length: $\boldsymbol{\pi}^{(1)} \cdot \mathbf{R} \cdot (\mathbf{I} - \mathbf{R})^{-2} \cdot \mathbf{1}^T$.

- Storage requirements for computation of queue length

  - No additional requirements.

Tables 5.3 and 5.4 summarize the computational and storage costs of the two methods.

Computation of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ (matrix geometric) or $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, $\boldsymbol{\pi}^{(*)}$ (Etaqa-GI/M/1)	
ETAQA-GI/M/1	$O^L(\eta\{\hat{\mathbf{L}}, \hat{\mathbf{F}}, \hat{\mathbf{B}}, \mathbf{L}, \mathbf{F}, \mathbf{B}, \mathbf{R}\}) + O(p \cdot (m \cdot \eta\{\hat{\mathbf{B}}, \mathbf{R}\} + n \cdot \eta\{\mathbf{B}, \mathbf{R}\}))$
Matrix-geometric	$O^L(\eta\{\hat{\mathbf{L}}, \hat{\mathbf{F}}, \hat{\mathbf{B}}, \mathbf{L}, \mathbf{B}, \mathbf{R}\}) + O(p \cdot (m \cdot \eta\{\hat{\mathbf{B}}, \mathbf{R}\} + n \cdot \eta\{\mathbf{B}, \mathbf{R}\})) + n^3)$
First moment measures	
ETAQA-GI/M/1	$O^L(\eta\{\mathbf{F}, \mathbf{L}, \mathbf{B}, \mathbf{R}\}) + O(p^2(m \cdot \eta\{\hat{\mathbf{B}}, \mathbf{R}\} + n \cdot \eta\{\mathbf{B}, \mathbf{R}\})$
Matrix-geometric	$O(n^2)$

Table 5.3: Computational complexity of the ETAQA-GI/M/1 solution and the matrix geometric method.

	Additional storage	**Storage of the probabilities**
Computation of $\boldsymbol{\pi}^{(0)}$ (matrix-geometric) or $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, $\boldsymbol{\pi}^{(*)}$ (Etaqa-GI/M/1)		
ETAQA-GI/M/1	$O(m \cdot n + n^2)$	$m + 2n$
Matrix-geometric	$O(m \cdot n + n^2)$	$m + n$
First moment measures		
ETAQA-GI/M/1	none	*none*
Matrix-geometric	none	*none*

Table 5.4: Storage complexity of the ETAQA-GI/M/1 solution and the matrix geometric method.

The two tables indicate that the classic matrix geometric method is the preferable method for solving GI/M/1-type processes. The major advantage of the matrix-geometric solution is its simplicity. The geometric relation between the vectors of the stationary probability distribution allows for simple closed form formulas for the computation of measures of interest such as the system queue length. Our ETAQA-GI/M/1 method may perform better when we are only interested in the computation of the probability vectors, depending on the system sparsity, the size of matrices, and the number of stored matrices that capture the behavior of the whole process but has higher complexity when used to compute measures of interest.

5.3 ETAQA-QBD

In Chapter 3, we noted that quasi-birth-death (QBD) processes are an intersection of M/G/1-type and GI/M/1-type processes while in Eq.(2.18) we defined the blocked-partitioned infinitesimal generator for a QBD process as follows

$$\mathbf{Q}_{QDB} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

QBDs can be solved with either matrix analytic or matrix-geometric method. Of the two methods, the method of choice for the solution of QBD processes, is matrix geometric because of its simplicity and ability to provide closed form formulas for measures of interest such as the average queue length. In contrary to the matrix analytic methods that solve QBDs using matrix geometric solution, we choose to solve QBDs using the ETAQA-M/G/1 since it is more efficient than ETAQA-GI/M/1.

The first step toward the solution of the QBD process is the computation of matrix $\mathbf{G}$. Recall that the logarithmic reduction algorithm [47], the most efficient algorithm for the computation of $\mathbf{R}$ requires first the computation of $\mathbf{G}$. Assuming the knowledge of matrix $\mathbf{G}$ for a QBD process with the infinitesimal generator as shown in Eq.(2.18), the proposed aggregate solution for the QBD process is stated from the following theorem:

Theorem Given an ergodic CTMC with infinitesimal generator $\mathbf{Q}_{QBD}$ having the structure shown in Eq.(2.18), with stationary probability vector $\boldsymbol{\pi} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, ...]$, the system of linear equations

$$\mathbf{x} \cdot \mathbf{X} = [1, \mathbf{0}], \tag{5.22}$$

where $\mathbf{X} \in \mathbb{R}^{(m+2n)\times(m+2n)}$ is defined as follows

$$\mathbf{X} = \left[\begin{array}{c|c|c|c} \mathbf{1}^T & \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0}^\circ \\ \mathbf{1}^T & \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^\circ \\ \mathbf{1}^T & \mathbf{0} & \mathbf{B} - \mathbf{F}\cdot\mathbf{G} & (\mathbf{L} + \mathbf{F} + \mathbf{F}\cdot\mathbf{G})^\circ \end{array}\right], \tag{5.23}$$

admits a unique solution $\mathbf{x} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$, where $\boldsymbol{\pi}^{(*)} = \sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)}$.

Proof. The steps in the proof are identical to the steps in the Proof of Theorem 5.1 since QBDs are special case of M/G/1-type processes. □

5.3.1 Computing measures of interest for QBD processes

Similarly to the M/G/1 case, our methodology allows the computation of the reward rate of state $s_i^{(j)}$, for $j \geq 2$ and $i = 1, \ldots, n$, if it is a polynomial of degree k in j with arbitrary coefficients $\mathbf{a}_i^{[0]}, \mathbf{a}_i^{[1]}, \ldots, \mathbf{a}_i^{[k]}$:

$$\forall j \geq 2,\ \forall i \in \{1, 2, \ldots, n\}, \qquad \rho_i^{(j)} = \mathbf{a}_i^{[0]} + \mathbf{a}_i^{[1]} j + \cdots + \mathbf{a}_i^{[k]} j^k.$$

Here, we follow the exact same steps as in Section 5.1.1, albeit significantly simplified. Observe that that $\mathbf{r}^{[0]}$ is simply $\boldsymbol{\pi}^{(*)}$ while, for $k > 0$, $\mathbf{r}^{[k]}$ can be computed after having obtained $\mathbf{r}^{[l]}$ for $0 \leq l < k$, by solving the system of n linear equations:

$$\begin{cases} \mathbf{r}^{[k]}(\mathbf{B} + \mathbf{L} + \mathbf{F})^{\circ} & = \mathbf{b}^{[k]\circ} \\ \mathbf{r}^{[k]}(\mathbf{F} - \mathbf{B})\mathbf{1}^T & = c^{[k]} \end{cases} . \tag{5.24}$$

where

$$\mathbf{b}^{[k]} = -\left(2^k \boldsymbol{\pi}^{(0)} \cdot \widehat{\mathbf{F}} + 2^k \boldsymbol{\pi}^{(1)} \cdot \mathbf{L} + 3^k \boldsymbol{\pi}^{(1)} \cdot \mathbf{F} + \sum_{l=1}^{k} \binom{k}{l} (2^l \mathbf{r}^{[k-l]} \cdot \mathbf{F} + \mathbf{r}^{[k-l]} \cdot \mathbf{L})\right)$$

and

$$c^{[k]} = -2^k \boldsymbol{\pi}^{(1)} \mathbf{F} \mathbf{1}^T - \sum_{l=1}^{k} \binom{k}{l} \mathbf{r}^{[k-l]} \cdot \mathbf{F} \cdot \mathbf{1}^T.$$

The rank of the system of linear equations depicted in Eq.(5.24) is n, since it QBD is a special case of M/G/1-type processes and we have discussed them in section 5.1.

We conclude by reiterating that in order to compute the k^{th} moment of the queue length we must solve k linear systems in n unknowns each and, in particular, the expected queue length is obtained by solving just one linear system in n unknowns.

5.3.2 Time and storage complexity

In this section, we present a detailed comparison of our aggregate solution for QBD processes with the matrix geometric method for QBDs. The notation in this section follows the one defined in section 5.1.2. We outline the required steps for each method and analyze the computation and storage complexity of each step up to the computation of the expected queue length. We assume that the algorithm of choice for computation of **R** in the matrix geometric solution for QBDs is logarithmic reduction as the most efficient one. Therefore in our analysis we do not include the cost to compute matrix **G**, which is the first matrix to be computed by logarithmic reduction [47].

In the following, we summarize the complexity analysis for ETAQA-QBD

- Computation of the aggregate stationary probability vector $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$
  - $O(n \cdot \eta\{\mathbf{F}, \mathbf{G}\})$ to compute **FG**,
  - $O^L(\widehat{\mathbf{L}}, \widehat{\mathbf{F}}, \widehat{\mathbf{B}}, \mathbf{B}, \mathbf{L}, \mathbf{F}, \mathbf{G})$ to solve the system of $m + 2n$ linear equations.
- Storage requirements for computation of $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$
  - $O(n^2)$ for matrix **FG**,
  - $m + 2n$ for the vector $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$.
- Computation of the queue length
  - $O(\eta\{\mathbf{F}, \mathbf{L}, \mathbf{B}\})$ to compute $\mathbf{F} + \mathbf{L} + \mathbf{B}$ and $\mathbf{F} - \mathbf{B}$.

  - $O^L(\eta\{\mathbf{F}, \mathbf{L}, \mathbf{B}\})$ to solve a system of n linear equations.

- Storage requirements for the computation of the queue length
  - No additional storage.

In the following, we summarize the complexity analysis for QBDs assuming that matrix-geometric is used for their solution:

- Computation of the boundary stationary probability vector $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}]$
  - $O(n^3)$ to compute $\mathbf{R}$ from $\mathbf{G}$ (last step of the logarithmic reduction algorithm) using the relation $\mathbf{R} = -\mathbf{F}(\mathbf{L} + \mathbf{FG})^{-1}$.
  - $O(n^3)$ to compute $(\mathbf{I} - \mathbf{R})^{-1}$,
  - $O(n \cdot \eta\{\mathbf{R}, \mathbf{B}\}$ to compute $\mathbf{RB}$,
  - $O^L(\widehat{\mathbf{L}}, \widehat{\mathbf{F}}, \widehat{\mathbf{B}}, \mathbf{L}, \mathbf{B}, \mathbf{R})$ for the solution of the system of $m+n$ linear equations to obtain $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$. The required storage for the probability vectors $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ is exactly $m + n$.
- Storage requirements to compute $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}]$
  - $O(n^2)$ for matrix $\mathbf{R}$ and $(\mathbf{I} - \mathbf{R})^{-1}$,
  - $m + n$ to store $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$.
- Computation of the queue length
  - $O(n^2)$ to compute queue length from $\boldsymbol{\pi}^{(1)} \cdot \mathbf{R} \cdot (\mathbf{I} - \mathbf{R})^{-2} \cdot \mathbf{1}^T$.

- **Storage requirements to compute queue length**
  - **No additional storage**

Tables 5.5 and 5.6 summarize the discussion in this section.

Computation of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ (matrix geometric) or $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, $\boldsymbol{\pi}^{(*)}$ (Etaqa-QBD)	
ETAQA-QBD	$O^L(\eta\{\widehat{\mathbf{L}}, \widehat{\mathbf{B}}, \widehat{\mathbf{F}}, \mathbf{L}, \mathbf{F}, \mathbf{B}, \mathbf{G}\}) + O(n \cdot \eta\{\mathbf{F}, \mathbf{G}\}))$
Matrix geometric	$O^L(\eta\{\widehat{\mathbf{L}}, \widehat{\mathbf{B}}, \widehat{\mathbf{F}}, \mathbf{L}, \mathbf{B}, \mathbf{R}\}) + O(n^3) + O(n \cdot \eta\{\mathbf{R}, \mathbf{B}\}$
First moment measures	
ETAQA-QBD	$O^L(\eta\{\mathbf{B}, \mathbf{L}, \mathbf{F}\}) + O(\eta(\mathbf{B}, \mathbf{L}, \mathbf{F}))$
Matrix geometric	$O(n^2)$

Table 5.5: Computational complexity of the ETAQA-QBD solution and the matrix geometric method.

	Additional storage	**Storage of the probabilities**
Computation of $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$ (matrix geometric) or $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, $\boldsymbol{\pi}^{(*)}$ (Etaqa-QBD)		
ETAQA-QBD	$O(n^2)$	$m + 2n$
Matrix geometric	$O(n^2)$	$m + n$
First moment measures		
ETAQA-QBD	none	*none*
Matrix geometric	none	*none*

Table 5.6: Storage complexity of the ETAQA-QBD solution and the matrix geometric method.

We emphasize the fact that the sparsity of $\mathbf{G}$ is key to preserving the sparsity of the original process in the ETAQA-QBD method, while the $\mathbf{R}$ that is required in matrix-geometric is usually full.

Concluding our analysis, we stress that the ETAQA-QBD solution is an efficient tool for solving QBD processes since it is easier to implement and results in significant gains with respect to computation. We note that even storage-wise we do gain (although this gain is not obvious using O-notation) because the aggregate solution requires only *temporal* storage of the matrix $\mathbf{F} \cdot \mathbf{G}$, while the matrix geometric method needs *persistent* storage of $\mathbf{R}$ and $(\mathbf{I} - \mathbf{R})^{-1}$.

5.4 Computational efficiency

In the previous section, we argue using O-notation about the the computational and storage efficiency of ETAQA-M/G/1. Here, we present further numerical evidence that ETAQA-M/G/1 is more efficient than other methods. For our comparisons, we use the classic Ramaswami's formula and the fast FFT implementation of Ramaswami's formula, the most efficient known algorithm for solving M/G/1-type processes [58]. We used Meini's implementation[1] for the cyclic reduction for the computation of $\mathbf{G}$ that is required in all three solution algorithms. We also used Meini's code for the fast FFT implementation of Ramaswami's formula that was made available to us via a personal communication [57]. We implemented the ETAQA-M/G/1 method and the classic Ramaswami's formula in C. All experiments were conducted on a 450 MHz Sun Enterprise 420R server with 4 GB memory.

The chain we selected for our experiments represents a general BMAP/M/1 queue. Recall that in practice, it is not possible to store an infinite number of $\widehat{\mathbf{F}}^{(i)}$ and $\mathbf{F}^{(i)}$ matrices, $1 < i < \infty$. One should stop storing when all entries of $\widehat{\mathbf{F}}^{(i)}$ and $\mathbf{F}^{(i)}$ for $i > p$ are below a sufficient threshold (i.e., *very close* to zero in a practical implemen-

[1]Code available at `http://www.dm.unipi.it/~meini/ric.html`.

tation) [47]. As illustrated in the previous section, computation time does depend on the size (i.e., parameters m and n) and the number (of stored) matrices (i.e., parameter p) that define the infinitesimal generator $\mathbf{Q}$. Finally, one last parameter that affects computation time is the number s of vector probabilities that should be computed so as the normalization condition $\sum_{i=1}^{s} \pi^{(i)} = 1.0$ is reached (again, within a sufficient numerical threshold).

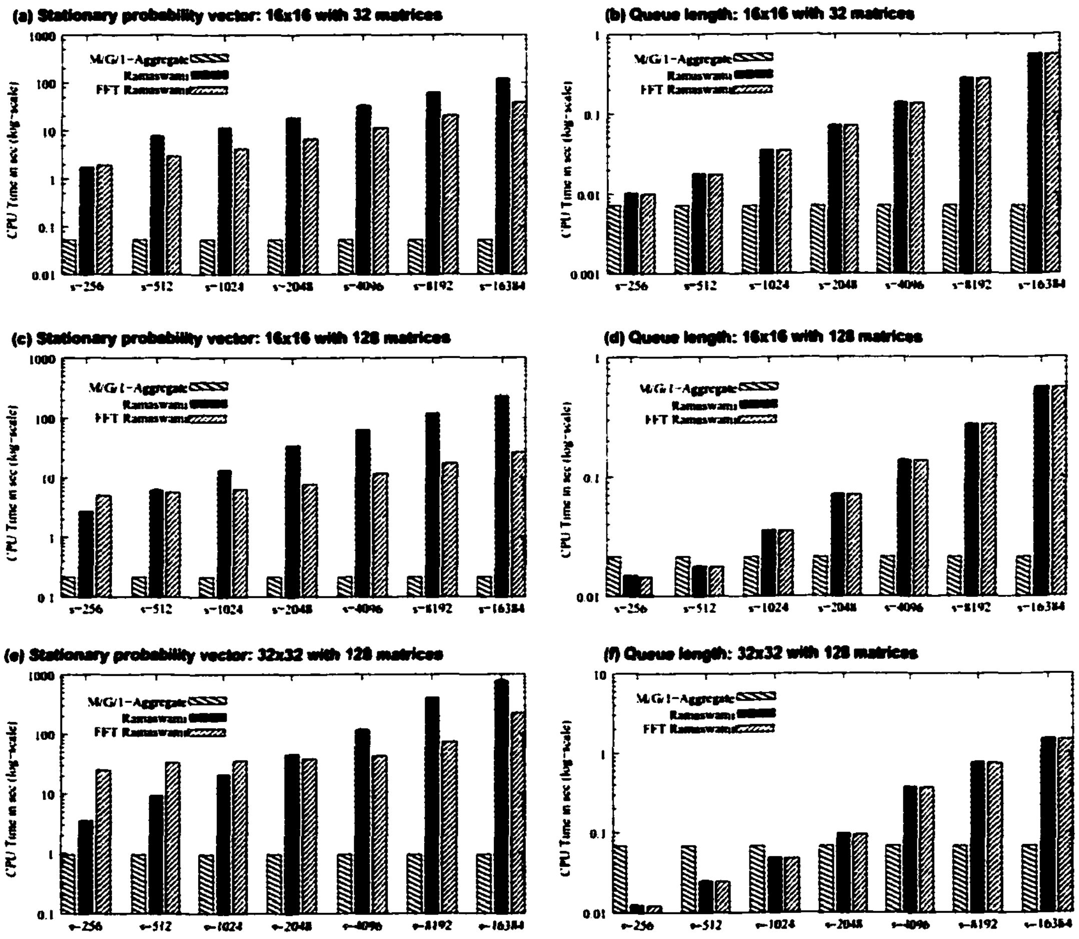

Figure 5.3: Execution time (in seconds) for ETAQA-M/G/1, the classic implementation of Ramaswami's formula, and the fast FFT implementation of Ramaswami's formula. The figure illustrates timings for the computation of the stationary probability vector (left column) and the computation of the queue length (right column).

In our experiments, we vary the parameters n, p, and s (for the case of BMAP/M/1 queue $m = n$) and provide timing results for the computation of the stationary vector $\boldsymbol{\pi}$ using the classic Ramaswami implementation and the fast FFT implementation, and the computation of $(\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)})$ using ETAQA-M/G/1. We also provide timings for the computation of the queue length for both methods. Results are presented in Figure 5.3.

The first experiment, considers a BMAP/M/1 queue with $n = 16$ and $p = 32$, a relatively small case. The timings[2] of the three algorithms are shown as a function of s. Figure 5.3(a) depicts the computation cost of the probability vector and Figure 5.3(b) illustrates the computation cost the queue length. Observe that the y-axis is in log-scale. Note that the value of s does affect the execution time of both Matrix-analytic approaches, but does not have any affect on ETAQA-M/G/1. As expected, for the computation of the stationary vector, the FFT implementation is superior to the classic Ramaswami formula, behavior that persists when we increase p and n (see Figures 5.3(c) and 5.3(e)). ETAQA-M/G/1 consistently outperforms the other two methods, plus its performance is insensitive to s (see figures Figures 5.3(a), 5.3(c) and 5.3(e)).

Figures 5.3(b), 5.3(d) and 5.3(f) illustrate the computation cost of the queue length for the three algorithms for various values of n, p, and s. Note that the two implementations of Ramaswami's formula have the same cost, since the same classic

[2]We point out that our timing results do not take into consideration the computation of $\mathbf{G}$, which is used in all three methods

formula is used for the computation of queue length: first weight appropriately and then sum the probability vector which is already computed. The figures further confirm that the cost of solving a small system of linear equations that ETAQA-M/G/1 requires for the computations of queue length is in many cases preferable to the classic methods. If this linear system increases and s is also small. then the classic methods may offer better performance.

5.5 Numerical stability of the method

The algorithm proposed in Section 5.1 results in a finite system of linear equations that can be solved with numerical methods. Because the linear system is a result of matrix additions, subtractions, and multiplications, its *numerical stability* should be examined. However, because of the presence of a linear system, and because our matrices are not M-matrices, an analytic examination of the numerical stability is not easily feasible. In this section we argue via experimentation that ETAQA-M/G/1 is numerically stable and compare its stability behavior with Ramaswami's recursive formula. Ramaswami's recursive formula for the computation of the steady state probability vector of an M/G/1-type process consists of matrix additions of non-negative matrices and these computations are known to be numerically stable [3].

In the following, we focus on the *stability of the method* used to solve the original

[3]We opt not to compare ETAQA-M/G/1 with the Fast FFT Ramaswami's formula because the FFTs may be the source of numerical instabilities [58].

problem, rather than the ***stability of the problem*** itself. The latter is measured by a condition number (or conditioning), which depends on a specific instance of the problem, but not on the method used.

The stability of a method $A : \Re^N \longrightarrow \Re^M$, given an input $x \in \Re^N$, is determined as follows:

$$||A(x) - A(x + \delta)|| < \kappa(x) \cdot ||\delta||$$

where $\delta \in \Re^N$ is a small perturbation of the input, and $\kappa(x)$ is the conditioning of the problem with input x. Any norms of the corresponding vector spaces can be used, but in the following we limit our discussion to the infinity (maximum) norm. [109] states that ***a stable algorithm gives nearly the right answer to nearly the right question.*** In other words, if we change the input of a stable algorithm by a ***small*** δ we should obtain an output that is perturbed, within the constant factor $\kappa(x)$, by a corresponding amount.

We follow the above definition to examine experimentally the stability of ETAQA-M/G/1 versus that of Ramaswami's formula. The output of the aggregate scheme is a probability vector of $m + 2n$ elements and is denoted as $A(x)$, where x belongs to the domain of the method, i.e., it is a choice of all the elements of the input matrices. The output of Ramaswami's is again a probability vector of $m+2n$ elements and is denoted as $R(x)$. Note that Ramaswami's original output is post-processed to produce the same aggregate probabilities that $A(x)$ produces. We run two sets of experiments, one for a well conditioned instance of the problem, and one for an ill-conditioned instance. This is performed via the following steps:

1. Select a CTMC and solve it using both ETAQA-M/G/1 and Ramaswami's formula and check the closeness of the results.

2. Perturb the input of the selected CTMC by *small* random numbers. We select three different perturbation magnitudes: 10^{-12}, 10^{-10} and 10^{-6}, and solve the CTMC with the perturbed input.

3. Repeat the perturbation experiment 50 times with different sets of random perturbation values within the same magnitude range to achieve statistical accuracy.

4. Calculate the perturbation of output for each randomly perturbed input for ETAQA-M/G/1 solutions considering as base case the output obtained by using ETAQA-M/G/1 to solve the CTMC without any perturbation of input, i.e., $\|A(x) - A(x + \delta_i)\|$, for each experiment i. Calculate the same for the set of perturbed solutions using Ramaswami's formula, where the base case is the solution obtained using Ramaswami's formula on the un-perturbed input, i.e., $\|R(x) - R(x + \delta_i)\|$, for each experiment i..

5. Calculate the absolute difference between the solution obtained by using ETAQA-M/G/1 and Ramaswami's formula, i.e., $\|A(x+\delta_i) - R(x+\delta_i)\|$, for each experiment i.

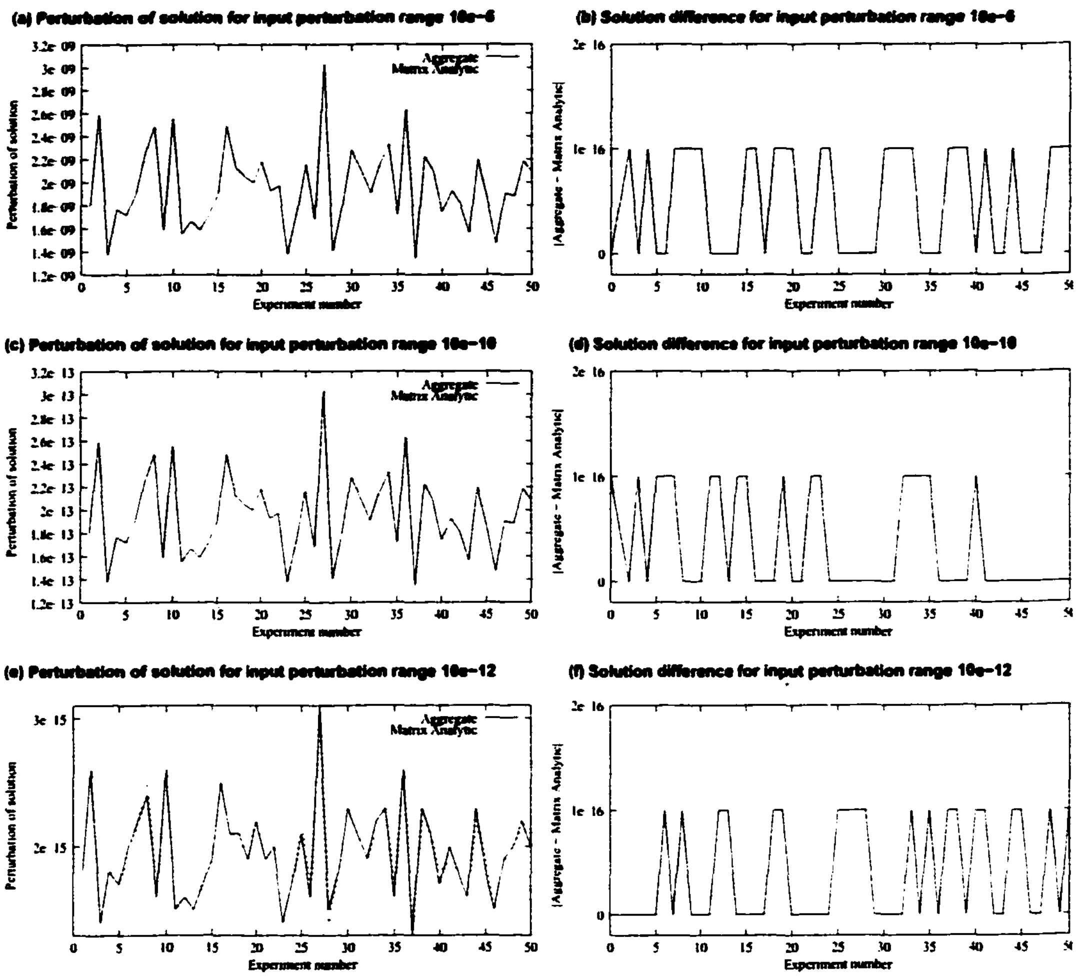

Figure 5.4: Numerical behavior of algorithms under well-conditioned input. Graphics (a), (c), and (e) plot $\|A(x) - A(x + \delta_i)\|$ for ETAQA-M/G/1 and $\|R(x) - R(x + \delta_i)\|$ for Ramaswami's recursive formula for different perturbation magnitudes and for $1 \leq i \leq 50$ distinct experiments. Graphics (b), (d), and (f) illustrate $\|A(x + \delta_i) - R(x + \delta_i)\|$ for the same perturbation magnitudes and the same $1 \leq i \leq 50$ distinct experiments.

For our experiments, we selected a CTMC that models a bursty hyper-exponential server with burst sizes ranging from 1 to $p = 64$. The dimension of matrices $\mathbf{B}$, $\mathbf{L}$ and $\mathbf{F}^{(i)}$ for $1 \leq i \leq p$ is 16×16 and matrices $\widehat{\mathbf{L}}$, $\widehat{\mathbf{B}}$ a $\widehat{\mathbf{F}}^{(i)}$ for $1 \leq i \leq p$ are of dimensions 1×1, 16×1 and 1×16 respectively. Since the corresponding $\mathbf{G}$ matrix for the process as well as matrices $\widehat{\mathbf{S}}^{(i)}$ and $\mathbf{S}^{(i)}$ for $1 \leq i \leq p$ are full, we consider the case to a representative one.[4] All experiments are conducted on a Pentium III with 64-bit double precision arithmetic, and 10^{-16} machine precision.

Our first set of experiments considers *well-conditioned* input matrices, where the values of their elements differ at most by two orders of magnitude. Figure 5.4 illustrates the behavior of ETAQA-M/G/1 and Ramaswami's formula under well-conditioned input for 50 distinct experiments. Each experiment corresponds to a different δ_i but within the same magnitude range. Figure 5.4(a) shows the perturbation of solution for each of 50 experiments for ETAQA-M/G/1 and Ramaswami's formula, is within the same magnitude range of 10^{-9}. Observe that Figure 5.4(a) does present two lines, one for ETAQA-M/G/1 and one for Ramaswami's formula but the lines are almost indistinguishable at this level. The proximity of the two solutions is better illustrated in Figure 5.4(b) where the difference between the solutions obtained by the two different methods is plotted and is in the range of $[0.0, 10^{-16}]$. The two methods are equal for all numerical purposes. Figures 5.4(c) and 5.4(e) illustrate the perturbation of solution for both methods with δ_i's in the range of 10^{-10} and 10^{-12}, respectively. Across the three experiments, the degree of perturbation in the solution (i.e., the conditioning of the problem) is within three orders of magnitude less than δ_i. Consistently with Figure 5.4(b), Figures 5.4(d) and 5.4(f) illustrate that the

[4]We conducted a large number of stability experiments but due to space restrictions we only present a few experiments here. We note however that *all* of our experiments did produce consistent results with those presented in this section.

two methods agree to machine precision. Regardless of the magnitude of the input perturbation, the differences between the solutions are *consistently* within the same range, i.e., 10^{-16}.

Next, we turn to a worse conditioned problem, where the elements within the various input matrices vary significantly. We use the same CTMC as the one in the previous set of experiments but the entries in all input matrices vary in magnitude up to 10^{11} with the largest element in the range of 10^{2} and the smallest in the range of 10^{-9}. Therefore, by increasing the stiffness of the problem the possibility of numerical errors increases. Again, we perturb the input with random values within ranges of 10^{-12}, 10^{-10}, and 10^{-6}. Results are presented in Figure 5.5.

The perturbation of input matrices with values at the level of 10^{-6} introduces a perturbation of the solution in the range of 10^{-7}, higher than the perturbation of solution in the well-conditioned case (compare Figures 5.5(a) and 5.4(a)). We point out that there are two lines on top of each-other in Figure 5.5(a) corresponding to ETAQA-M/G/1 and Ramaswami's output respectively. The differences between the solutions obtained by both methods for each experiments are presented in Figure 5.5(b) and are in the range of $[0.0, 1.8 \times 10^{-14})$. Comparing to the results of the well-conditioned case we note an increase on the difference among the two solutions, but still very small and clearly less than the perturbation value. Figures 5.5(c) and 5.5(e) illustrate the perturbation of solutions for perturbation of inputs in the ranges of 10^{-10} and 10^{-12} respectively. Comparing to the results of Figure 5.4, we observe that the

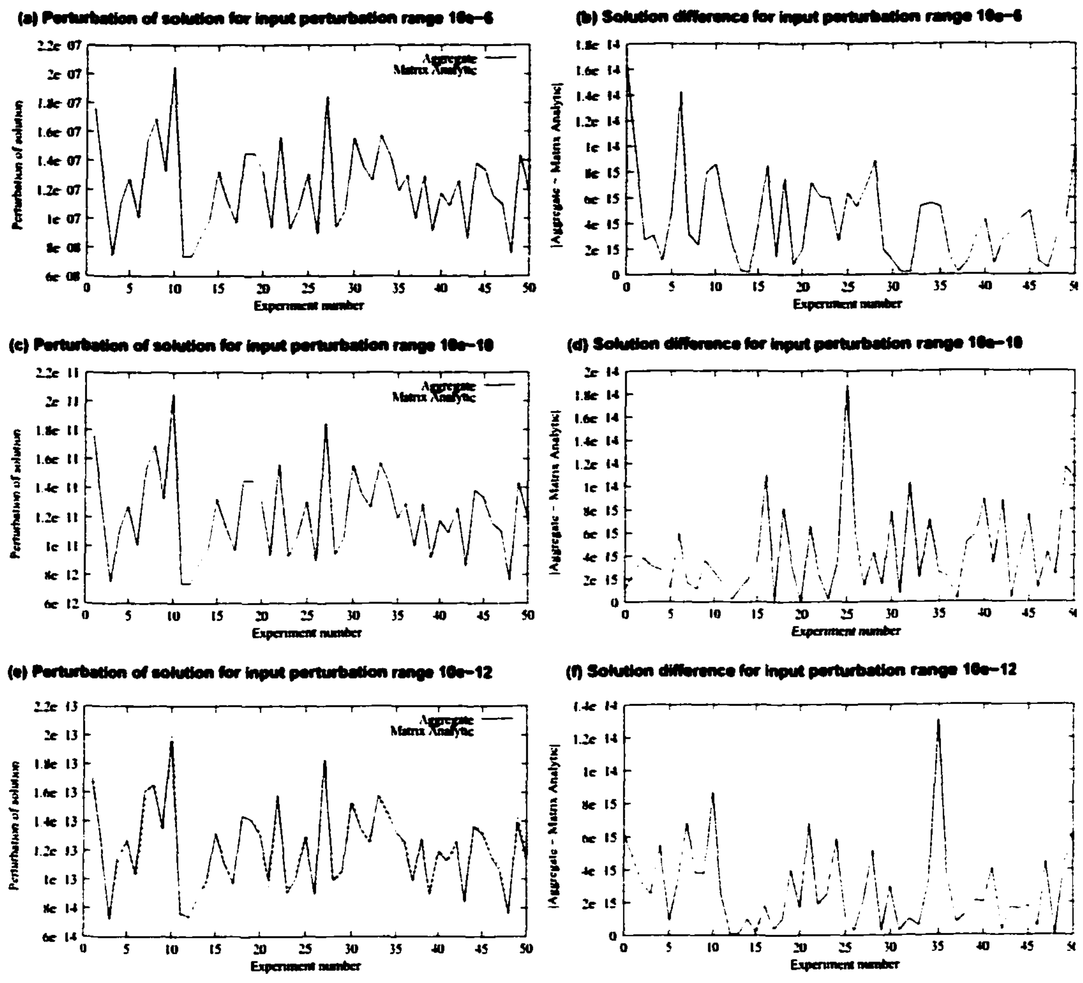

Figure 5.5: Numerical behavior of the two algorithms with ill-conditioned input. Graphics (a), (c), and (e) plot $\|A(x) - A(x+\delta_i)\|$ for ETAQA-M/G/1 and $\|R(x) - R(x+\delta_i)\|$ for Ramaswami's recursive formula for different perturbation magnitudes and for $1 \le i \le 50$ distinct experiments. Graphics (b), (d), and (f) illustrate $\|A(x+\delta_i) - R(x+\delta_i)\|$ for the same perturbation magnitudes and the same $1 \le i \le 50$ distinct experiments.

conditioning of the problem increases. The degree of perturbation remains constant for all three experiments and is one order of magnitude less than δ_i, consistently across experiments. The difference of solutions between the two methods in the case of input perturbation ranges of 10^{-10} and 10^{-12} are presented in Figures 5.5(d) and

5.5(f). The differences are within the same range as for the experiment depicted in Figure 5.5(b).

The results presented in Figures 5.4 and 5.5 show that both methods, ETAQA-M/G/1 and Ramaswami's formula behave very similarly under different numerical scenarios. Since for nearly the same input we obtain, in both cases, nearly the same output, we argue that the stability of Ramaswami's recursive formula is re-confirmed. Our experiments also illustrate that ETAQA-M/G/1 and Ramaswami's recursive formula are in good agreement.

5.6 MAMSolver

MAMSOLVER is a software tool that provides efficient implementations of the state-of-the-art algorithms of the matrix-analytic methodology including the matrix-analytic and matrix-geometric solution techniques for M/G/1-type and GI/M/1-type processes, respectively. MAMSOLVER also provides an implementation of the ETAQA methodology. Although, this exposition of matrix-analytic methods and ETAQA considers only CTMCs, MAMSOLVER provides solutions for both DTMCs and CTMCs.

The matrix-analytic algorithms that we take into consideration are defined in terms of matrices, making matrix manipulations and operations the basic elements of the tool. The input to MAMSolver, in the form of a structured text file, is the finite set of matrices that accurately describe the process to be solved. Since there are different algorithms that provide solution for the same process, the user specifies the method to be used. However, several tests are performed within the tool to insure that special cases are treated separately and therefore more efficiently. MAMSOLVER is implemented in C++, and classes define the basic components of the type of processes under consideration.

Matrix is the module that implements all the basic matrix operations such as matrix assignments, additions, subtractions, multiplications, and inversions. For computational efficiency, we use well known and heavily tested routines provided by the Lapack and BLAS packages[5]. Since solving a finite system of linear equations is a core function in matrix-analytic algorithms, MAMSolver provides several numerical methods depending on the size of the problem, i.e., the size of the coefficient matrices. For small-size problems exact methods such as LU-decomposition are used. otherwise the Lapack implementation of iterative methods such as GMRES and BiCGSTAB. are chosen.

Matrix-analytic modules handle both CTMC and DTMC processes. First these modules provide storage for the input matrices. In addition, these modules provide all the routines necessary for the implementation of the algorithms outlined in Chapter 3. Both the data structures and routines of the matrix-analytic modules are based on the data-structures and routines provided by the matrix module. The high-level structure of MAMSOLVER is illustrated in Figure 5.6.

The solution of **QBD** processes, requires computation of the **R** (and sometimes

[5]Available from http://www.netlib.org.

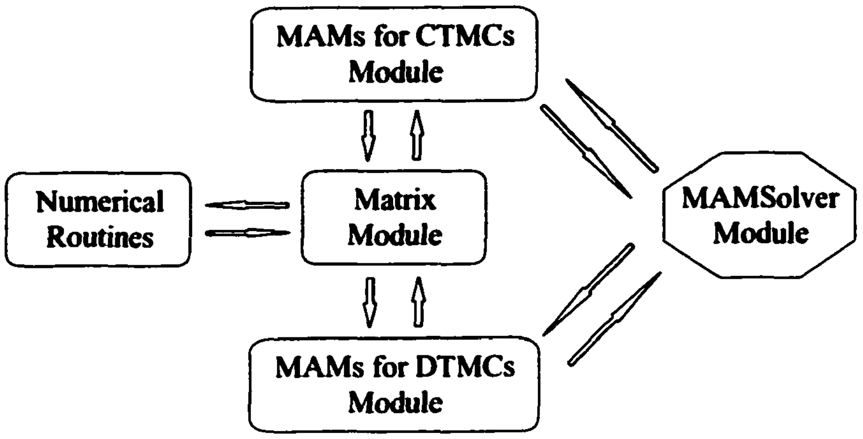

Figure 5.6: MAMSolver structure

of **G** depending on the solution algorithm). First the matrix **R** is computed using the logarithmic reduction algorithm[47]. For completeness, we provide also the classic numerical algorithm. To guarantee that there is no expensive (and unnecessary) iterative computation of **G** (and **R**), the tool first checks if the conditions for explicit computation hold [77]. The available solution methods for QBD processes are matrix-geometric and ETAQA-QBD.

GI/M/1 processes require the computation of the matrix **R**. The classic matrix geometric solution is implemented to solve this type of processes. First the algorithm goes through a classic iterative algorithm to compute **R** (to our knowledge, there is no alternative more efficient than the classic algorithm). Then, the tool computes the boundary part of the stationary probability vector. Since a geometric relation exist between vectors $\boldsymbol{\pi}^{(i)}$ for $i \geq 1$, there is no need to compute the whole stationary probability vector.

M/G/1 processes require the computation of the matrix **G** which is calculated using the classic iterative algorithm or the cyclic-reduction algorithm or the explicit one (if applied). The stationary probability vector is computed recursively using either the recursive Ramaswami formula or its fast FFT version. ETAQA-M/G/1 is also implemented as an alternative for the solution of M/G/1 processes. A brief summary of the most important matrix-analytic algorithms implemented in MAMSOLVER is depicted in Figure 5.7.

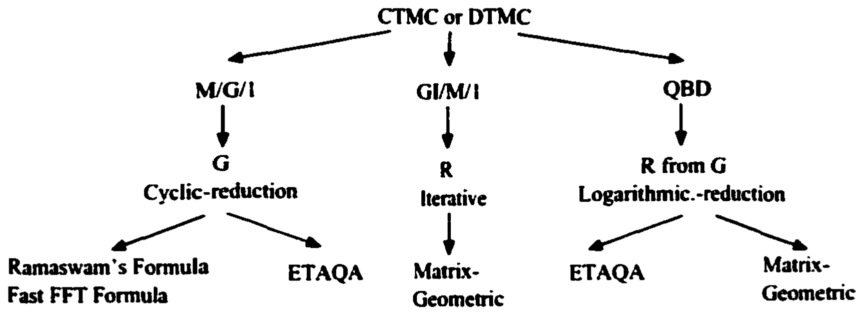

Figure 5.7: MAMSolver algorithms

5.6.1 MAMSolver input and output

The input to MAMSOLVER is stored in an ASCII file and given via redirection. All input matrices are stored by rows and they include even the zero entries. The input file has the following structure:

- size m of the boundary portion of the process,
- size n of the repeating portion of the process,
- number l of matrices that accurately define the infinite set of matrices that describe the repeating portion of the process,

- desired solution accuracy ϵ,
- the matrix that describes the interaction among states of the boundary portion, $\widehat{\mathbf{L}}$,
- the matrix that describes the interaction between states of the boundary portion and states of the first inter-level for GI/M/1 ($\widehat{\mathbf{F}}$) and vice-versa for QBD and M/G/1 processes ($\widehat{\mathbf{B}}$),
- the matrices that describe the interaction between states of the i^{th} inter-level and states of the boundary portion for GI/M/1 processes ($\widehat{\mathbf{B}}^{(i)}$) and vice-versa for QBD and M/G/1 processes ($\widehat{\mathbf{F}}^{(i)}$); there are l such matrices,
- for GI/M/1-type processes only, the matrices that correspond to the interaction between the first inter-level and the rest of the repeating portion of the process; there are $l + 1$ such matrices,
- the matrix that describes the forward ($\mathbf{F}$) or backward ($\mathbf{B}$) transitions in the repeating portion of the GI/M/1 or QBD and M/G/1 processes respectively,
- the matrix that describes the local transitions in the repeating portion of the process ($\mathbf{L}$),
- the matrices that describe the backward ($\mathbf{B}^{(i)}$) or forward transition $\mathbf{F}^{(i)}$ in the repeating portion of GI/M/1 or QBD and M/G/1 processes respectively; there are l such matrices.

The output from MAMSOLVER consist of several ASCII files as follows:

- "probability" stores the stationary probability vector, $\boldsymbol{\pi}$. For a QBD or GI/M/1 process it stores only $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$ because the rest of $\boldsymbol{\pi}$ is computed using **R** and $\boldsymbol{\pi}^{(1)}$. For an M/G/1 process the entire $\boldsymbol{\pi}$ is stored in file "probability".
- "R-matrix" stores the matrix **R** if a QBD or GI/M/1 process is solved.
- "G-matrix" stores the matrix **G** if a QBD or M/G/1 process is solved.
- "QL-length" stores the average queue length from the solution of the queueing system. Momentarily MAMSolver computes a customized average queue length, i.e., first moment of $\boldsymbol{\pi}$, of the form:

$$r = \boldsymbol{\pi}^{(0)} \cdot \boldsymbol{\rho}^{(0)} + \boldsymbol{\pi}^{(1)} \cdot \boldsymbol{\rho}^{(1)} + \sum_{i=2}^{infty} \boldsymbol{\pi}^{(i)}(\mathbf{a}^{[0]} + \mathbf{a}^{[1]} \cdot i).$$

  The coefficient vectors $\boldsymbol{\rho}^{(0)}$, $\boldsymbol{\rho}^{(1)}$, $\mathbf{a}^{[0]}$, $\mathbf{a}^{[1]}$ are to be read from the text file "queue". In case of failure to open file "queue" their default values are: $\boldsymbol{\rho}^{(0)} = \mathbf{0}$, $\boldsymbol{\rho}^{(1)} = \mathbf{1}$, $\mathbf{a}^{[0]} = \mathbf{0}$, $\mathbf{a}^{[1]} = \mathbf{1}$. Computation of higher moments can be achieved by introducing additional coefficient vectors $\mathbf{a}^{[j]}$ in the above formula.
- "QL-dist" stores the queue length distribution of the queueing system. "QL-dist" is a two-column file where the first column represents the values of the queue length and the second column their respective steady-state probabilities of occurrence.

- "drift-conditions" stores the queue stability condition and the value of the caudal characteristic for QBD processes.

Examples of the input to MAMSOLVER for simples Markov chains are given in Appendix D. The entire source code for MAMSOLVER is available at the tool's website: http://www.cs.wm.edu/~riska/MAMSolver.html.

5.7 Chapter summary

In this chapter, we presented a new aggregate approach for the exact solution of M/G/1-type, GI/M/1-type, and QBD processes. Our exposition focuses on computing efficiently the *exact* probabilities of the boundary states of the process and the aggregate probability distribution in each of the classes of states corresponding to a specific partitioning of the remaining infinite portion of the state space. Although the method does not compute the probability distribution of all states, it still provides enough information for the "mathematically exact" computation of a wide variety of Markov reward functions such as the expected queue length or any of its higher moments.

We presented detailed analysis of the computation and storage complexity of our method. We conclude that for the case of M/G/1-type and QBD processes our methodology requires a few simple steps that provide significant savings with respect to both computation and storage when compared with the traditional matrix analytic and matrix geometric methods, respectively. For M/G/1 processes our methodology results in a much simpler. thus easy to implement, algorithm comparing to matrix analytic. We showed, via numerical experiments, that ETAQA is not only computational efficient but also numerically stable. For the case of GI/M/1-type processes, our methodology has the same complexity as the classic matrix geometric method for the computation of the stationary probability vector. albeit the classic method results in less expensive and more intuitively appealing formulas for the computation of measures of interest such as the expected queue length.

Chapter 6

Aggregate Solutions of GI/G/1-type Processes

In this chapter, we study a class of CTMCs with GI/G/1-type patterns in their repetitive structure, that is they exhibit *both* M/G/1-type and GI/M/1-type patterns. GI/G/1-type processes cannot be solved exactly by existing techniques, but can alternatively be approximated by QBDs [34]. Such processes occur when modeling open systems that accept customers from multiple exogenous sources (i.e., accepting bulk arrivals) and are also subject to failures and repairs (i.e., the system may empty-out in a single step when a catastrophic failure occurs or only parts of it may be operational when a non-catastrophic failure occurs).

Recall that, as defined in Eq.(2.23), the infinitesimal generator **Q** can be block-partitioned as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{L}^{(0)} & \widehat{\mathbf{F}}^{(1)} & \widehat{\mathbf{F}}^{(2)} & \widehat{\mathbf{F}}^{(3)} & \widehat{\mathbf{F}}^{(4)} & \cdots \\ \widehat{\mathbf{B}}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \widehat{\mathbf{B}}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \widehat{\mathbf{B}}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \widehat{\mathbf{B}}^{(4)} & \mathbf{B}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \qquad (6.1)$$

Our approach focuses on separating the GI/M/1-type behavior of the system from the M/G/1-type behavior by defining new decomposed CTMCs using stochastic complementation and pseudo-stochastic complementation techniques. The decomposed CTMCs of the M/G/1 and GI/M/1 type can be solved using matrix analytic methods. The solution of the original GI/G/1-type CTMC is then obtained by coupling the solutions of the decomposed CTMCs of M/G/1-type and GI/M/1-type.

The chapter is organized as follows. In Section 6.1, we outline some additional results on stochastic complementation. In Section 6.2, we define pseudo-stochastic complementation as a variation of stochastic complementation. We give the high level idea of our approach in Section 6.3. The general approach is outlined in Section 6.4, followed by formalization of the algorithm in Sections 6.4, 6.5, 6.6, 6.7, 6.8. The case when multiple M/G/1-type and GI/M/1-type subchains can be identified within the original GI/G/1-type CTMC is analyzed in Section 6.9. In Section 6.10, the applicability of the technique is illustrated by solving a GI/G/1-type CTMC that arises in the area of parallel processor scheduling. We conclude the chapter with a summary of the results.

6.1 Non-disjoint coupling

In this section, we further discuss stochastic complementation, introduced in Subsection 2.8.2. As defined in [61], to apply stochastic complementation in a Markov chain its state space $\mathcal{S}$ is partitioned into two *disjoint* subsets, $\mathcal{A}$ and $\overline{\mathcal{A}}$. Observe that, if the stationary probability vector, $\boldsymbol{\alpha}$, of the stochastic complement of states in $\mathcal{A}$ is known, we can compute the stationary distribution of states in a subset $\mathcal{B}$ of $\mathcal{A}$ conditioned on being in $\mathcal{B}$, as:

$$\boldsymbol{\beta} = \frac{\boldsymbol{\alpha}[\mathcal{B}]}{\boldsymbol{\alpha}[\mathcal{B}] \cdot \mathbf{1}^T}, \tag{6.2}$$

and this allows us to extend the concept of stochastic complementation to cases where we use a covering, not a partitioning, of the state space, that is, $\mathcal{S} = \mathcal{A}_1 \cup \mathcal{A}_2$, but the subsets $\mathcal{A}_1$ and $\mathcal{A}_2$[1] are not necessarily disjoint. This is motivated by the fact that, in certain cases, it may be advantageous to compute the stochastic complement for subsets of states that are not disjoint (see Lemma 2.8.2).

Definition (Singular states) Consider a covering of the state space $\mathcal{S} = \mathcal{A}_1 \cup \mathcal{A}_2$. Any state $h \in \mathcal{S}$ that belongs to more than one subset in the covering is said to be a *singular* state.

Lemma (Non-disjoint coupling) Consider an ergodic CTMC with state space $\mathcal{S}$ covered by the subsets $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ be their stationary probability vectors for the respective stochastic complement,

[1]We use indices in the case of stochastic complementation on non-disjoint subsets to distinguish from the case when the subsets are disjoint

also assumed to be ergodic. Let s be the number of singular states in the covering, and assume, without loss of generality, that the sets $\mathcal{B}_1$ and $\mathcal{B}_2$ of non-singular states in $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively are not empty. Compute $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, the stationary probabilities for the states in $\mathcal{B}_1$ and $\mathcal{B}_2$ conditioned on being in $\mathcal{B}_1$ and $\mathcal{B}_2$ for the stochastic complements of states in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, as in Eq.(6.2). Define a new "coupled" CTMC with state space given by two macro states, for $\mathcal{B}_1$ and $\mathcal{B}_2$, plus the s singular states. For ease of notation, let $\mathcal{B}_{2+i}$ denote the set containing the i^{th} singular state *only* and let its conditional stationary probability $\beta_{2+i} = 1$, for $i = 1, \ldots, s$. Let the infinitesimal generator matrix $\mathbf{C}$ of this CTMC be:

$$\mathbf{C}[i,j] = \boldsymbol{\beta}_i \cdot \mathbf{Q}[\mathcal{B}_i, \mathcal{B}_j] \cdot \mathbf{1}^T. \tag{6.3}$$

Then, the stationary probability vector $\mathbf{a}$ of this CTMC gives the correct coupling factors corresponding to this covering, that is (after an appropriate reordering of the entries in $\boldsymbol{\pi}$):

$$\mathbf{a}[i] = \boldsymbol{\pi}[\mathcal{B}_i] \cdot \mathbf{1}^T, \; i = 1, \ldots, 2+s \tag{6.4}$$

which then implies

$$\boldsymbol{\pi} = [\mathbf{a}[1] \cdot \boldsymbol{\beta}_1, \mathbf{a}[2] \cdot \boldsymbol{\beta}_2, \mathbf{a}[2], \ldots, \mathbf{a}[2+s]]. \tag{6.5}$$

Proof: From its definition, it follows that the off-diagonal elements of $\mathbf{C}$ are non-negative and its rows sum to zero because the diagonal elements are defined to be the negative of the row sums.

Matrix $\mathbf{C}$ is clearly irreducible, this follows directly from its probabilistic interpretation and the fact that the original process is irreducible. Assuming that $\mathbf{C}$ is also aperiodic, hence ergodic, we now need to show that the stationary probability vector $\mathbf{a} = [\mathbf{a}[1], \ldots, \mathbf{a}[s+2]]$ satisfies Eq.(6.4). Since the stationary probability vector $\boldsymbol{\alpha}_i$ of the stochastic complement of $\mathcal{A}_i$ satisfies $\boldsymbol{\alpha}_i = \boldsymbol{\pi}[\mathcal{A}]/(\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T)$ $i = 1, 2$, we get that

$$\beta_i = \frac{\boldsymbol{\alpha}_i[\mathcal{B}_i]}{\boldsymbol{\alpha}_i[\mathcal{B}_i] \cdot \mathbf{e}} = \frac{\boldsymbol{\pi}[\mathcal{B}_i]/(\boldsymbol{\pi}[\mathcal{A}_i] \cdot \mathbf{e})}{\boldsymbol{\pi}[\mathcal{B}_i]/(\boldsymbol{\pi}[\mathcal{A}_i] \cdot \mathbf{e}) \cdot \mathbf{e}} = \frac{\boldsymbol{\pi}[\mathcal{B}_i]}{\boldsymbol{\pi}[\mathcal{B}_i] \cdot \mathbf{e}} \tag{6.6}$$

while $\beta_i = 1 = \boldsymbol{\pi}[\mathcal{B}_i]/(\boldsymbol{\pi}[\mathcal{B}_i] \cdot \mathbf{e})$ by definition for $i = 3, \ldots, 2+s$. We want to prove that the stationary probability vector $\mathbf{a}$ of the CTMC with infinitesimal generator $\mathbf{C}$ is $[\boldsymbol{\pi}[\mathcal{B}_1] \cdot \mathbf{e}, \ldots, \boldsymbol{\pi}[\mathcal{B}_{k+s}] \cdot \mathbf{e}]$ and so Eq.(6.5) holds. For $j = 1, \ldots, 2+s$, recalling that $\boldsymbol{\pi} \cdot \mathbf{Q} = \mathbf{0}$, we have:

$$\begin{aligned}
(\mathbf{aC})[i] &= \sum_{j=1}^{2+s} \mathbf{a}[j] \cdot \mathbf{C}[j,i] \\
&= \sum_{j=1}^{2+s} \boldsymbol{\pi}[\mathcal{B}_j] \cdot \mathbf{1}^T \cdot \beta_i \cdot \mathbf{Q}[\mathcal{B}_i, \mathcal{B}_j] \cdot \mathbf{1}^T \\
&= \sum_{j=1}^{2+s} \boldsymbol{\pi}[\mathcal{B}_j] \cdot \mathbf{1}^T \cdot \frac{\boldsymbol{\pi}[\mathcal{B}_i]}{\boldsymbol{\pi}[\mathcal{B}_i] \cdot \mathbf{1}^T} \cdot \mathbf{Q}[\mathcal{B}_i, \mathcal{B}_j] \cdot \mathbf{1}^T \\
&= \sum_{j=1}^{2+s} \boldsymbol{\pi}[\mathcal{B}_i] \cdot \mathbf{Q}[\mathcal{B}_i, \mathcal{B}_j] \cdot \mathbf{1}^T = 0
\end{aligned}$$

Thus, $[\pi[\mathcal{B}_1]\cdot \mathbf{e}, \ldots, \pi[\mathcal{B}_{k+s}]\cdot \mathbf{e}]$ is a solution of $\mathbf{a}\cdot\mathbf{C} = \mathbf{0}$, and, since its entries sum to one, it is the stationary probability vector of the coupled CTMC. □

It is important to note that the effectiveness of stochastic complementation is based on divide-and-conquer: computing the stationary distribution of each stochastic complement of the original CTMC must be easier than computing the stationary distribution of the original CTMC. Computing the product $\mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}](-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]$ in Eq.(2.28) can be costly since $(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}}])^{-1}$ is effectively an inverse, thus often full. However, there are cases where we can take advantage of the special structure of the CTMC and short-circuit this computation such as the "single-entry" case as defined and described in Lemma 2.8.2 of Subsection 2.8.2.

6.2 Pseudo-stochastic complementation

Theorem (Pseudo-stochastic complement) Given an ergodic CTMC with infinitesimal generator matrix $\mathbf{Q}$ and stationary probability vector π, whose state space is partitioned into two disjoint sets $\mathcal{A}$ and $\overline{\mathcal{A}}$ the pseudo-stochastic complement of $\mathcal{A}$ as is defined as:

$$\mathbf{A} = \mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}]\cdot\mathbf{1}^T\cdot Norm(\overline{\alpha}\cdot\mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]), \tag{6.7}$$

where $\overline{\alpha} = \pi[\overline{\mathcal{A}}]/(\pi[\overline{\mathcal{A}}]\cdot\mathbf{1}^T)$ is the stationary conditional distribution of the states in $\overline{\mathcal{A}}$ for the original CTMC. Then, the CTMC with transition probability matrix $\mathbf{A}$ is irreducible with stationary distribution $\boldsymbol{\alpha} = \boldsymbol{\pi}[\mathcal{A}]/(\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T)$.

Proof: It is easy to see that $\mathbf{A}$ is an infinitesimal generator: its off-diagonal entries are non-negative by construction, and its rows sum to zero because:

$$\begin{aligned}
\mathbf{A} \cdot \mathbf{1}^T = &= \left(\mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T \cdot Norm(\boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}])\right) \cdot \mathbf{1}^T \\
&= \mathbf{Q}[\mathcal{A},\mathcal{A}] \cdot \mathbf{1}^T + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T \cdot Norm(\boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]) \cdot \mathbf{1}^T = \\
&= \mathbf{Q}[\mathcal{A},\mathcal{A}] \cdot \mathbf{1}^T + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T = \\
&= \mathbf{Q}[\mathcal{A},\mathcal{S}] \cdot \mathbf{1}^T = \mathbf{0} \qquad (6.8)
\end{aligned}$$

That the pseudo stochastic complement of $\mathcal{A}$ is irreducible follows directly from its probabilistic interpretation and the fact that the original process is irreducible. We now need to show that $\boldsymbol{\alpha} \cdot \mathbf{A} = \mathbf{0}$, when $\boldsymbol{\alpha} = \boldsymbol{\pi}[\mathcal{A}]/(\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T)$. Starting from the definition of pseudo stochastic complement, we have:

$$\begin{aligned}
\boldsymbol{\alpha} \cdot \mathbf{A} &= \frac{\boldsymbol{\pi}[\mathcal{A}]}{\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T} \cdot \left(\mathbf{Q}[\mathcal{A},\mathcal{A}] + \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T \cdot Norm(\boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}])\right) \\
&= \frac{1}{\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T} \cdot \left(\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{Q}[\mathcal{A},\mathcal{A}] + \boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T \cdot \frac{\boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]}{\boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T}\right) \\
&= \frac{1}{\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{1}^T} \cdot \left(\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{Q}[\mathcal{A},\mathcal{A}] + \boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}]\right) = \mathbf{0}. \qquad (6.9)
\end{aligned}$$

where the last two steps are obtained considering that $\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T = \boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\mathcal{A},\overline{\mathcal{A}}] \cdot \mathbf{1}^T$, (they represent the flow from $\mathcal{A}$ to $\overline{\mathcal{A}}$ and from $\overline{\mathcal{A}}$ to $\mathcal{A}$, respectively) and that $\boldsymbol{\pi}[\mathcal{A}] \cdot \mathbf{Q}[\mathcal{A},\mathcal{A}] + \boldsymbol{\pi}[\overline{\mathcal{A}}] \cdot \mathbf{Q}[\overline{\mathcal{A}},\mathcal{A}] = \mathbf{0}$ (since $\boldsymbol{\pi}$ is the stationary probability vector). This completes the proof. □

Comparing the definitions of stochastic and pseudo-stochastic complementation, we see that the former is naturally based on considering all paths starting and ending in $\mathcal{A}$ (given the interpretation of the inverse as an infinite sum over all possible paths), while the latter makes use of the conditional stationary probability vector for $\overline{\mathcal{A}}$. What is interesting though, is that, even if the two complements are described by different matrices, Theorems 2.8.2 and 6.2 imply that they *have the same stationary probability vector* $\boldsymbol{\alpha}$. The intuition behind this property is given by the stochastic meaning of the matrix $(-\mathbf{Q}[\overline{\mathcal{A}},\overline{\mathcal{A}})^{-1}$ used in the definition of stochastic complement. Its entry in row r and column c represents the expected amount of time spent in state $c \in \overline{\mathcal{A}}$ starting from state $r \in \overline{\mathcal{A}}$ before entering $\mathcal{A}$, a quantity that is of course related to the stationary probability vector $\boldsymbol{\pi}[\overline{\mathcal{A}}]$, used in the formulation of the pseudo-stochastic complement.

A way to use both stochastic and pseudo-stochastic complementation is as follows. If we can locate one subset of states $\mathcal{A}$ in $\mathcal{S}$ having a single entry state, we can apply the idea of stochastic complementation, obtain a smaller CTMC corresponding only to $\mathcal{A}$, and solve it for its stationary distribution, which is then the stationary conditional distribution of $\mathcal{A}$ in $\mathcal{S}$. Then we apply pseudo-stochastic complementation over the complementary state space $\overline{\mathcal{A}}$, and analogously find its stationary conditional distribution in $\mathcal{S}$. The computation is efficient as it consists only of matrix multiplications that preserve the sparsity of the original matrices.

6.3 GI/G/1 solution (High-level idea)

In this section, we present the high-level idea of our algorithm and outline the attributes of the structure of the CTMCs that we can solve. We observe that the pattern of interaction among states of a CTMC with infinitesimal generator $\mathbf{Q}$ given by Eq.(6.1) is the union of the patterns for GI/M/1-type and M/G/1-type processes, thus it is more general than either. Based on this observation, we propose the following solution steps:

1. We partition the union of the level sets $\mathcal{S} = \bigcup_{j=1}^{\infty} \mathcal{S}^{(j)}$, into two disjoint sets $\mathcal{U}$ and $\mathcal{L}$ such that $\mathcal{U}$ captures the GI/M/1-type behavior of $\mathbf{Q}$ and $\mathcal{L}$ captures the M/G/1-type behavior of $\mathbf{Q}$. We elaborate on the exact meaning of this statement in Section 6.4.

2. We use the well-known concept of stochastic complementation [61] to define two new processes (stochastic complements), one containing all states in $\mathcal{U}$ (plus a finite number of states $\mathcal{G}_g^+ \subseteq \mathcal{S}^{(0)}$) and one containing all states in $\mathcal{L}$ (plus a finite number of states $\mathcal{G}_g^- \subseteq \mathcal{S}^{(0)}$).

3. We solve each new process using well-known techniques, and obtain the conditional stationary probabilities for all states in $\mathcal{U} \cup \mathcal{G}_g^+$ (or $\mathcal{L} \cup \mathcal{G}_g^-$) given that the original process $\mathbf{Q}$ is in $\mathcal{U} \cup \mathcal{G}_g^+$ (or $\mathcal{L} \cup \mathcal{G}_g^-$, respectively). In particular, the stochastic complement of the set $\mathcal{U} \cup \mathcal{G}_g^+$ is a GI/M/1-type process that is

solved with the matrix geometric method [67] and the stochastic complement[2] of $\mathcal{L} \cup \mathcal{G}_g^-$ is an M/G/1-type process that is solved using the matrix analytic method [69, 47].

4. Finally, we "couple" the two solutions in order to scale back the conditional stationary probabilities of all states in $\mathcal{U} \cup \mathcal{G}_g^+$ and $\mathcal{L} \cup \mathcal{G}_g^-$ and obtain the stationary probabilities of the original process.

Figure 6.1 illustrates the main structure of the CTMC required by the above steps, while the corresponding block structure of **Q** is shown in Table 6.1. We observe a "two-level" repetitive structure, where each level set $\mathcal{S}^{(j)}$, $j \geq 1$, is partitioned into two disjoint classes denoted $\mathcal{U}^{(j)}$ (for "upper") and $\mathcal{L}^{(j)}$ (for "lower"). For the moment, we opt not to discuss any partitioning of the boundary portion of the process $\mathcal{S}^{(0)}$. Let $\mathcal{U} = \bigcup_{j=1}^{\infty} \mathcal{U}^{(j)}$ and $\mathcal{L} = \bigcup_{j=1}^{\infty} \mathcal{L}^{(j)}$. The following list summarizes the interactions within each set and across the two sets.

- Within set $\mathcal{U}$, forward transitions are allowed from any $\mathcal{U}^{(j)}$ only toward the next level $\mathcal{U}^{(j+1)}$. Backward transitions are allowed from set $\mathcal{U}^{(j)}$ to any lower level sets $\mathcal{U}^{(k)}$, $k < j$.

- Within set $\mathcal{L}$, forward transitions are allowed from any $\mathcal{L}^{(j)}$ toward any higher level $\mathcal{L}^{(k)}$, $k > j$. Backward transitions are allowed from set $\mathcal{L}^{(j)}$ to $\mathcal{L}^{(j-1)}$ only.

- Arbitrary local transitions (not shown) are allowed within each $\mathcal{U}^{(j)}$ and $\mathcal{L}^{(j)}$.

[2]Actually we apply the pseudo stochastic complement to generate the M/G/1 process with set of states $\mathcal{L} \cup \mathcal{G}_g^-$.

- Transitions from $\mathcal{U}^{(j)}$ to any $\mathcal{L}^{(k)}$, $k \geq 1$ are allowed.
- There is strictly no interaction from $\mathcal{L}$ toward $\mathcal{U}$ (except, of course, through the boundary portion $\mathcal{S}^{(0)}$).
- There is a special "gate" state g in $\mathcal{S}^{(0)}$ such that any path from $\mathcal{L}$ to $\mathcal{U}$ must visit state g. In practice, such a gate might exist in $\mathcal{S}^{(1)}$ but not in $\mathcal{S}^{(0)}$; in this case, we simply redefine a new $\mathcal{S}^{(0)}$ as the union of the original sets $\mathcal{S}^{(0)}$ and $\mathcal{S}^{(1)}$ and "shift all levels to the left by one".

The gated structure for the interaction of $\mathcal{U}$ and $\mathcal{L}$ is critical for our algorithm, as it allows us to apply stochastic complementation in a special setting (see Subsection 2.8.2). Furthermore, in conjunction with the upper/lower interaction between sets $\mathcal{U}$ and $\mathcal{L}$, it ensures that (a) the stochastic complement of the upper set of states is a GI/M/1-type CTMC and (b) the pseudo stochastic complement of the lower set of states is a M/G/1-type CTMC. As it will be explained in Subsection. 6.4, the actual algorithm can recursively apply to chains where multiple upper or lower sets are identified. We elaborate on this topic in Subsection 6.9.

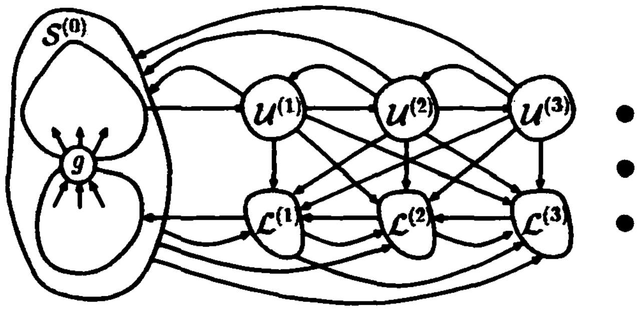

Figure 6.1: The two-level interaction between states and the gated structure of our CTMC.

	$\mathcal{U}^{(0)}$	$\overline{\mathcal{L}}^{(0)}$	$\mathcal{U}^{(1)}$	$\mathcal{L}^{(1)}$	$\mathcal{U}^{(2)}$	$\mathcal{L}^{(2)}$	$\mathcal{U}^{(3)}$	$\mathcal{L}^{(3)}$	$\dots$
$\mathcal{U}^{(0)}$	$\widetilde{\mathbf{L}}^{(0)}_{\mathcal{U}\mathcal{U}}$	$\widetilde{\mathbf{L}}^{(0)}_{\mathcal{U}\overline{\mathcal{L}}}$	$\widetilde{\mathbf{F}}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\widetilde{\mathbf{F}}^{(1)}_{\mathcal{U}\mathcal{L}}$		$\widetilde{\mathbf{F}}^{(2)}_{\mathcal{U}\mathcal{L}}$		$\widetilde{\mathbf{F}}^{(3)}_{\mathcal{U}\mathcal{L}}$	
$\overline{\mathcal{L}}^{(0)}$		$\widehat{\mathbf{L}}^{(0)}_{\overline{\mathcal{L}}\mathcal{L}}$		$\widehat{\mathbf{F}}^{(1)}_{\overline{\mathcal{L}}\mathcal{L}}$		$\widehat{\mathbf{F}}^{(2)}_{\overline{\mathcal{L}}\mathcal{L}}$		$\widehat{\mathbf{F}}^{(3)}_{\overline{\mathcal{L}}\mathcal{L}}$	
$\mathcal{U}^{(1)}$	$\mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(1)}_{\mathcal{U}\overline{\mathcal{L}}}$	$\mathbf{L}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{L}^{(1)}_{\mathcal{U}\mathcal{L}}$	$\mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}}$		$\mathbf{F}^{(2)}_{\mathcal{U}\mathcal{L}}$	
$\mathcal{L}^{(1)}$		$\widehat{\mathbf{B}}^{(1)}_{\mathcal{L}\overline{\mathcal{L}}}$		$\mathbf{L}^{(1)}_{\mathcal{L}\mathcal{L}}$		$\mathbf{F}^{(1)}_{\mathcal{L}\mathcal{L}}$		$\mathbf{F}^{(2)}_{\mathcal{L}\mathcal{L}}$	
$\mathcal{U}^{(2)}$	$\mathbf{B}^{(2)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(2)}_{\mathcal{U}\overline{\mathcal{L}}}$	$\mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(1)}_{\mathcal{U}\mathcal{L}}$	$\mathbf{L}_{\mathcal{U}\mathcal{U}}$	$\mathbf{L}_{\mathcal{U}\mathcal{L}}$	$\mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}}$	
$\mathcal{L}^{(2)}$				$\widehat{\mathbf{B}}^{(1)}_{\mathcal{L}\mathcal{L}}$		$\mathbf{L}_{\mathcal{L}\mathcal{L}}$		$\mathbf{F}^{(1)}_{\mathcal{L}\mathcal{L}}$	
$\mathcal{U}^{(3)}$	$\mathbf{B}^{(3)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(3)}_{\mathcal{U}\overline{\mathcal{L}}}$	$\mathbf{B}^{(2)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(2)}_{\mathcal{U}\mathcal{L}}$	$\mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}}$	$\mathbf{B}^{(1)}_{\mathcal{U}\mathcal{L}}$	$\mathbf{L}_{\mathcal{U}\mathcal{U}}$	$\mathbf{L}_{\mathcal{U}\mathcal{L}}$	
$\mathcal{L}^{(3)}$						$\widehat{\mathbf{B}}^{(1)}_{\mathcal{L}\mathcal{L}}$		$\mathbf{L}_{\mathcal{L}\mathcal{L}}$	
$\vdots$									

Table 6.1: The nonzero pattern in our matrix **Q**.

6.4 GI/G/1 solution - General approach

We now outline and then describe in detail the new approach we propose for the stationary study of a CTMC with infinitesimal generator **Q** having the structure of (6.1), provided certain conditions apply. The step-by-step outline of our algorithm is:

(Upper-lower aggregation of Q)

Step 1: **Determine the upper and lower classes.**

The partition is defined by splitting each $\mathcal{S}^{(j)}$ into an upper set $\mathcal{U}^{(j)}$ and a lower set $\mathcal{L}^{(j)}$, consistent with a partition of the set of state indices $\mathcal{N} = \{1, \dots, n\}$ within a level set $\mathcal{S}^{(j)}$ into $\mathcal{N}_u$ and $\mathcal{N}_l$, that is:

$$\forall j \geq 1, \quad \mathcal{U}^{(j)} = \left\{s_i^{(j)} : i \in \mathcal{N}_u\right\} \quad \mathcal{L}^{(j)} = \left\{s_i^{(j)} : i \in \mathcal{N}_l\right\}.$$

Let $\mathcal{U} = \bigcup_{j=1}^{\infty} \mathcal{U}^{(j)}$ and $\mathcal{L} = \bigcup_{j=1}^{\infty} \mathcal{L}^{(j)}$. The key conditions this partition must satisfy are (refer to Figure 6.1):

- There is a special "gate" state $g \in \mathcal{S}^{(0)}$ such that any path from states in $\mathcal{L}$ to states in $\mathcal{U}$ must visit state g. Let $\mathcal{G}^-$ be the set of states in $\mathcal{S}^{(0)}$ that can be reached from $\mathcal{L}^{(1)}$ without visiting g first, and let $\mathcal{G}_g^- = \mathcal{G}^- \cup \{g\}$. Let $\mathcal{G}^+ = \mathcal{S}^{(0)} \setminus \mathcal{G}_g^-$ and $\mathcal{G}_g^+ = \mathcal{G}^+ \cup \{g\}$, i.e., g is not included in neither $\mathcal{G}^-$ nor $\mathcal{G}^+$, but it is included in both $\mathcal{G}_g^-$ and $\mathcal{G}_g^+$.
- $\mathbf{Q}[\mathcal{G}_g^+ \cup \mathcal{U}, \mathcal{G}_g^+ \cup \mathcal{U}]$, is of the GI/M/1-type (or a special case of it, such as QBD).
- $\mathbf{Q}[\mathcal{G}_g^- \cup \mathcal{L}, \mathcal{G}_g^- \cup \mathcal{L}]$ is of the M/G/1-type (or a special case of it, such as QBD).
- The submatrix $\mathbf{Q}[\mathcal{U}, \mathcal{L}]$ can contain nonzero entries, that is, it might be possible to go from upper states to lower states.
- All other submatrices are zero. In particular, $\mathbf{Q}[\mathcal{L}, \mathcal{U}] = \mathbf{0}$, that is, there are no transitions from lower states to upper states. These states can communicate only through the boundary states.

If any of the above conditions is not satisfied by the partition, then we cannot apply the following steps of our algorithm. If instead the conditions are satisfied, then we proceed with the following steps.

Step 2: **Generate and solve the "upper CTMC".**

Using stochastic complementation, define a new CTMC observing the behavior of the original CTMC only in the states of $\mathcal{G}_g^+ \cup \mathcal{U}$. This is a GI/M/1-type

CTMC, so we can solve for steady state using the matrix geometric method.

Step 3: **Generate and solve the "lower CTMC".**

Generate the pseudo-stochastic complement corresponding to the states in $\mathcal{G}_g^- \cup \mathcal{L}$. This is an M/G/1-type CTMC, so we can solve it for steady state using the matrix analytic method.

Step 4: **Compute the coupling factors.**

At this point, we know the conditional stationary probabilities of all states in $\mathcal{G}_g^+ \cup \mathcal{U}$ and $\mathcal{G}_g^- \cup \mathcal{L}$. Note that state g appears at both $\mathcal{G}_g^+ \cup \mathcal{U}$ and $\mathcal{G}_g^- \cup \mathcal{L}$. Therefore, g is considered to be a singular state and we apply Lemma 6.1 in order to derive the coupling factors of $\{g\}$, $\mathcal{G}^+ \cup \mathcal{U}$ and $\mathcal{G}^- \cup \mathcal{L}$.

Step 5: **Generate the stationary distribution of the original process.**

The expressions for the stationary probability of the original process can then be obtained by multiplying the conditional stationary probabilities of all states in $\mathcal{G}^+ \cup \mathcal{U}$ and $\mathcal{G}^- \cup \mathcal{L}$ by their respective coupling factors.

In the following subsections, we describe in detail steps 1, 2, and 3.

6.5 Determining the upper and lower sets

Our methodology is based on decomposition, where, informally, we relegate the GI/M/1-type behavior into the upper CTMC and the M/G/1-type behavior into the lower CTMC. Thus the first step is the definition of these CTMCs in such a way that, first of all, they are individually solvable by known methods, and just as important, the results of their analysis are meaningful within the original process. We stress that we can achieve a GI/M/1-type structure in the upper CTMCs and a M/G/1-type structure in the lower CTMCs even if, in the original CTMC, each upper state can have arbitrarily long forward transition and backward transitions to lower states.

The upper CTMC is obtained by considering $\mathcal{U}$ together with the set $\mathcal{G}_g^+$. We eliminate all other states by using stochastic complementation, hence the single entry condition must be satisfied if we are to do so efficiently: this is the reason for requiring the existence of the gate g and for all transitions between $\mathcal{U}$ and $\mathcal{L}$ to exist *only* from $\mathcal{U}$ toward $\mathcal{L}$. Given these requirements, the blocks composing $\mathbf{Q}$ in Eq.(6.1) can be decomposed accordingly to the upper and lower sets. For example, if we have a single upper and a single lower set, we have:

$$\widehat{\mathbf{F}}^{(j)} = \begin{bmatrix} \mathbf{0} & \widehat{\mathbf{F}}^{(j)}_{\mathcal{L}} \end{bmatrix}, \mathbf{F}^{(1)} = \begin{bmatrix} \mathbf{F}^{(1)}_{\mathcal{UU}} & \mathbf{F}^{(1)}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{F}^{(1)}_{\mathcal{LL}} \end{bmatrix}, \mathbf{F}^{(j)} = \begin{bmatrix} \mathbf{0} & \mathbf{F}^{(j)}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{F}^{(j)}_{\mathcal{LL}} \end{bmatrix}, \quad j > 1.$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{\mathcal{UU}} & \mathbf{L}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{L}_{\mathcal{LL}} \end{bmatrix}, \mathbf{L}^{(1)} = \begin{bmatrix} \mathbf{L}^{(1)}_{\mathcal{UU}} & \mathbf{L}^{(1)}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{L}^{(1)}_{\mathcal{LL}} \end{bmatrix}$$

$$\widehat{\mathbf{B}}^{(j)} = \begin{bmatrix} \widehat{\mathbf{B}}^{(j)}_{\mathcal{U}} \\ \mathbf{0} \end{bmatrix}, \mathbf{B}^{(1)} = \begin{bmatrix} \mathbf{B}^{(1)}_{\mathcal{UU}} & \mathbf{B}^{(1)}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{B}^{(1)}_{\mathcal{LL}} \end{bmatrix}, \mathbf{B}^{(j)} = \begin{bmatrix} \mathbf{B}^{(j)}_{\mathcal{UU}} & \mathbf{B}^{(j)}_{\mathcal{UL}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad j > 1.$$

We could instead partition the original CTMC into U upper subchains and L lower chains that are still compliant to the basic requirement that all transitions between the subchains exist only from the upper to the lower chains, and, in addition, the corresponding diagonal blocks (i.e., $\mathbf{F}^{(1)}_{\mathcal{UU}}$, $\mathbf{F}^{(j)}_{\mathcal{CC}}$, $\mathbf{L}_{\mathcal{UU}}$, $\mathbf{L}_{\mathcal{CC}}$, $\mathbf{B}^{(1)}_{\mathcal{CC}}$, and $\mathbf{B}^{(j)}_{\mathcal{UU}}$, for $j \geq 1$) are themselves block diagonal. We return to this issue in Subsection 6.9, where we present a recursive version of our "upper-lower aggregation" algorithm, that deals efficiently with this case.

No restrictions exist for the off-diagonal blocks (i.e., $\mathbf{F}^{(1)}_{\mathcal{UC}}$, $\mathbf{F}^{(j)}_{\mathcal{UC}}$, $\mathbf{L}_{\mathcal{UC}}$, $\mathbf{B}^{(1)}_{\mathcal{UC}}$, and $\mathbf{B}^{(j)}_{\mathcal{UC}}$, for $j > 1$). Further restrictions in the structure of $\mathbf{L}^{(0)}$, $\widehat{\mathbf{F}}^{(1)}$, and $\widehat{\mathbf{B}}^{(1)}$, exist to reflect the requirement of being able to find the gate state and are discussed in the following section.

6.6 Determining a gate state

A gate state g for set $\mathcal{U}$ must exist among the states in $\mathcal{S}^{(0)}$ (as discussed earlier, if we do not find such a state in $\mathcal{S}^{(0)}$ but we find it in $\mathcal{S}^{(1)}$, we can shift all levels by one toward the boundary portion, with the only drawback that the size of $\mathcal{S}^{(0)}$ grows by an additional n states). Figure 6.2 shows the essential structure of the relevant portion of the infinitesimal generator, $\mathbf{Q}[\mathcal{S}^{(0)} \cup \mathcal{S}^{(1)}, \mathcal{S}^{(0)} \cup \mathcal{S}^{(1)}]$. We already know that the definition of the upper and lower sets implies that block $\mathbf{Q}[\mathcal{L}^{(1)}, \mathcal{U}^{(1)}]$ is zero. We now discuss the additional structure imposed by the need to have a gate state g for $\mathcal{U}$.

Consider the partition of $\mathcal{S}^{(0)}$ into $\mathcal{G}^+_g$, which of course includes g, and $\mathcal{G}^-$. Any state in $\mathcal{G}^+_g$ can have transitions to and from any state in $\mathcal{U}^{(1)}$. States in $\mathcal{G}^+_g$ can also have transitions to $\mathcal{G}^-$ and to $\mathcal{L}^{(1)}$, but only g can have transitions from them.

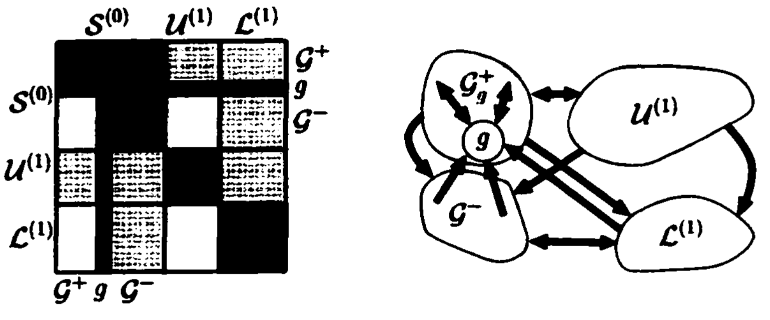

Figure 6.2: The structure implied on **Q** by the existence of the gate g.

Finally, $\mathcal{U}^{(1)}$ can have transitions to $\mathcal{G}^-$ (and to $\mathcal{L}^{(1)}$, of course), but it cannot have transitions from them.

6.7 Generation of the new "upper" process

The connectivity of $\mathcal{U}$ with $\mathcal{L}$ coupled with the existence of the gate state g ensures that there is an efficient way to obtain the stochastic complement of $\mathcal{U}$ by applying Lemma 2.8.2. The state space of the new process is $\mathcal{G}_g^+ \cup \mathcal{U}$. The infinitesimal generator of this new stochastic process is:

$$\mathbf{Q}_{\mathcal{G}_g^+\cup\mathcal{U}} = \begin{bmatrix} \mathbf{L}^{(0)}_{\mathcal{G}_g^+\mathcal{G}_g^+} & \widehat{\mathbf{F}}^{(1)}_{\mathcal{G}_g^+\mathcal{U}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(10)}_{\mathcal{U}\mathcal{G}_g^+} & \mathbf{L}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(20)}_{\mathcal{U}\mathcal{G}_g^+} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{L}_{\mathcal{U}\mathcal{U}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}}^{(30)}_{\mathcal{U}\mathcal{G}_g^+} & \mathbf{B}^{(2)}_{\mathcal{U}\mathcal{U}} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{L}_{\mathcal{U}\mathcal{U}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}} & \cdots \\ \widehat{\mathbf{B}}^{(40)}_{\mathcal{U}\mathcal{G}_g^+} & \mathbf{B}^{(3)}_{\mathcal{U}\mathcal{U}} & \mathbf{B}^{(2)}_{\mathcal{U}\mathcal{U}} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{U}} & \mathbf{L}_{\mathcal{U}\mathcal{U}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{6.10}$$

Matrices $\mathbf{F}^{(1)}_{\mathcal{U}\mathcal{U}}$, $\mathbf{L}_{\mathcal{U}\mathcal{U}}$, and $\mathbf{B}^{(j)}_{\mathcal{U}\mathcal{U}}$, $j \geq 1$, are defined in Section 6.5. $\mathbf{L}^{(0)}_{\mathcal{G}_g^+\mathcal{G}_g^+}$ contains the rows and columns of $\mathbf{L}^{(0)}$ corresponding to states in $\mathcal{G}_g^+$, except that the diagonal element $\mathbf{L}^{(0)}_{\mathcal{G}_g^+\mathcal{G}_g^+}[g, g]$ might differ from the corresponding one in $\mathbf{L}^{(0)}$, since any transition from g to $\mathcal{L}$ is rerouted back to g itself, hence ignored. $\widehat{\mathbf{F}}^{(1)}_{\mathcal{G}_g^+\mathcal{U}}$ contains the entries of $\widehat{\mathbf{F}}^{(1)}$ that represent the transitions from states in $\mathcal{G}_g^+$ to $\mathcal{U}$. What remains to be defined are the matrices $\widehat{\mathbf{B}}^{(j0)}_{\mathcal{U}\mathcal{G}_g^+}$, $j \geq 1$.

Since g is a single entry state for $\mathcal{U}$, all transitions from the states in $\mathcal{U}$ to $\mathcal{L}$ are "folded back" into g. Let

$$\widehat{\mathbf{B}}^{(j0)}_{\mathcal{U}\mathcal{G}_g^+} = \widehat{\mathbf{B}}^{(j)}_{\mathcal{U}\mathcal{G}_g^+} + \overline{\mathbf{B}}^{(j)}, \quad j \geq 1,$$

where $\widehat{\mathbf{B}}^{(j)}_{\mathcal{U}\mathcal{G}_g^+}$ represents the portion of $\widehat{\mathbf{B}}^{(j)}_{\mathcal{U}}$ corresponding to transitions from $\mathcal{U}$ to $\mathcal{G}_g^+$. $\overline{\mathbf{B}}^{(j)}$ is a matrix of zeros except for the gth column, which is obtained by applying Eq.(2.31) and Lemma 2.8.2:

$$\overline{\mathbf{B}}^{(j)}[\mathcal{N}_u, g] = \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)}_{\mathcal{U}\mathcal{L}} + \mathbf{L}_{\mathcal{U}\mathcal{L}} + \widehat{\mathbf{B}}^{(j)}_{\mathcal{U}\mathcal{G}^-} + \sum_{l=1}^{j-1} \mathbf{B}^{(l)}_{\mathcal{U}\mathcal{L}} \right) \cdot \mathbf{e} \quad j \geq 1,$$

where $\mathbf{e}$ is a row vector of one with the appropriate dimension.

The result is a GI/M/1-type process that is solved using the matrix-geometric method outlined in Section 3.1. Note that the computation of the sum $\sum_{j=1}^{\infty} \overline{\mathbf{B}}^{(j)}$ is not required: since $\overline{\mathbf{B}}^{(j)}$ is a matrix that contains only one nonzero column that is added to the gth column of $\widehat{\mathbf{B}}^{(j)}_{\mathcal{U}\mathcal{G}_g^+}$, we opt to discard this particular column when using Eq.(3.3).

After applying the matrix-geometric method, we obtain an expression for the stationary probabilities of the states in $\mathcal{G}_g^+ \cup \mathcal{U}$ conditioned on the original process being in any of these states. These conditional probabilities are needed to formulate the pseudo-stochastic complement of $\mathcal{G}_g^- \cup \mathcal{L}$ according to Theorem 6.2, as explained in the following section.

6.8 Generation of the new "lower" process

After applying Theorem 6.2, the infinitesimal generator of the new process with state space $\mathcal{G}_g^- \cup \mathcal{L}$ is given by:

$$\mathbf{Q}_{\mathcal{G}_g^- \cup \mathcal{L}} = \begin{bmatrix} \mathbf{L}^{(00)}_{\mathcal{G}_g^-\mathcal{G}_g^-} & \widehat{\mathbf{F}}^{(01)}_{\mathcal{G}_g^-\mathcal{L}} & \widehat{\mathbf{F}}^{(02)}_{\mathcal{G}_g^-\mathcal{L}} & \widehat{\mathbf{F}}^{(03)}_{\mathcal{G}_g^-\mathcal{L}} & \widehat{\mathbf{F}}^{(04)}_{\mathcal{G}_g^-\mathcal{L}} & \cdots \\ \widehat{\mathbf{B}}^{(1)}_{\mathcal{L}\mathcal{G}_g^-} & \mathbf{L}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(2)}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(3)}_{\mathcal{L}\mathcal{L}} & \cdots \\ \mathbf{0} & \mathbf{B}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{L}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(2)}_{\mathcal{L}\mathcal{L}} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{L}_{\mathcal{L}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{L}\mathcal{L}} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}^{(1)}_{\mathcal{L}\mathcal{L}} & \mathbf{L}_{\mathcal{L}\mathcal{L}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (6.11)$$

Matrices $\mathbf{F}^{(j)}_{\mathcal{L}\mathcal{L}}$ for $j \geq 1$, $\mathbf{L}^{(1)}_{\mathcal{L}\mathcal{L}}$, $\mathbf{L}_{\mathcal{L}\mathcal{L}}$, and $\mathbf{B}^{(1)}_{\mathcal{L}\mathcal{L}}$ are defined in Section 6.5. $\widehat{\mathbf{B}}^{(1)}_{\mathcal{L}\mathcal{G}_g^-}$ contains the entries of $\widehat{\mathbf{B}}^{(1)}$ that correspond to transitions from $\mathcal{G}_g^-$ to $\mathcal{L}$. What remains to be defined are the matrices $\widehat{\mathbf{F}}^{(0j)}_{\mathcal{G}_g^-\mathcal{L}}$, $j \geq 1$, and $\mathbf{L}^{(00)}_{\mathcal{G}_g^-\mathcal{G}_g^-}$.

Since all interaction of the "lower" process with states in $\mathcal{U}$ is done through $\mathcal{G}_g^-$, it follows that only the rates out of g need to be altered. Indeed, by applying Theorem 6.2 we see:

$$\mathbf{L}^{(00)}_{\mathcal{G}_g^-\mathcal{G}_g^-} = \mathbf{L}^{(0)}_{\mathcal{G}_g^-\mathcal{G}_g^-} + \overline{\mathbf{L}}$$

and

$$\widehat{\mathbf{F}}^{(0j)}_{\mathcal{G}_g^-\mathcal{L}} = \widehat{\mathbf{F}}^{(j)}_{\mathcal{L}\mathcal{L}} + \overline{\mathbf{F}}^{(j)}, \quad j \geq 1.$$

The new terms $\overline{\mathbf{L}}$ and $\overline{\mathbf{F}}^{(j)}$ are introduced by pseudo-stochastic complementation and represent how states from $\mathcal{G}^+$ and $\mathcal{U}$ enter the new process, respectively.

Let $\overline{\boldsymbol{\alpha}} = [\overline{\boldsymbol{\alpha}}^{(0)}, \overline{\boldsymbol{\alpha}}^{(1)}, \ldots]$ be conditional stationary distribution of $\mathcal{G}^+ \cup \mathcal{U}$. This can be derived by normalizing the stationary distribution of the stochastic complement of $\mathcal{G}_g^+ \cup \mathcal{U}$. Since the stochastic complement of the "upper" process is solved with the matrix geometric method, it follows that

$$\overline{\boldsymbol{\alpha}}^{(j)} = \overline{\boldsymbol{\alpha}}^{(1)} \cdot \mathbf{R}^{(j-1)} \quad j \geq 1.$$

Recall that the application of pseudo-stochastic complementation adds only a component $\mathbf{Q}[\mathcal{G}_g^- \cup \mathcal{L}, \mathcal{G}^+ \cup \mathcal{U}] \cdot \mathbf{e} \cdot Norm(\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}])$ to the first row-block of $\mathbf{Q}_{\mathcal{G}_g^- \cup \mathcal{L}}$. Since we allow transitions from any level in $\mathcal{U}$ to *all* levels in $\mathcal{L}$, matrix $\mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}]$ can be full:

$$\mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}] = \begin{bmatrix} \mathbf{L}^{(0)}_{\mathcal{G}^+\mathcal{G}_g^-} & \widehat{\mathbf{F}}^{(1)}_{\mathcal{G}^+\mathcal{L}} & \widehat{\mathbf{F}}^{(2)}_{\mathcal{G}^+\mathcal{L}} & \widehat{\mathbf{F}}^{(3)}_{\mathcal{G}^+\mathcal{L}} & \widehat{\mathbf{F}}^{(4)}_{\mathcal{G}^+\mathcal{L}} & \widehat{\mathbf{F}}^{(5)}_{\mathcal{G}^+\mathcal{L}} & \cdots \\ \widehat{\mathbf{B}}^{(1)}_{\mathcal{U}\mathcal{G}_g^-} & \mathbf{L}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(2)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(3)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(4)}_{\mathcal{U}\mathcal{L}} & \cdots \\ \widehat{\mathbf{B}}^{(2)}_{\mathcal{U}\mathcal{G}_g^-} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{L}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(2)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(3)}_{\mathcal{U}\mathcal{L}} & \cdots \\ \widehat{\mathbf{B}}^{(3)}_{\mathcal{U}\mathcal{G}_g^-} & \mathbf{B}^{(2)}_{\mathcal{U}\mathcal{L}} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{L}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(2)}_{\mathcal{U}\mathcal{L}} & \cdots \\ \widehat{\mathbf{B}}^{(4)}_{\mathcal{U}\mathcal{G}_g^-} & \mathbf{B}^{(3)}_{\mathcal{U}\mathcal{L}} & \mathbf{B}^{(2)}_{\mathcal{U}\mathcal{L}} & \mathbf{B}^{(1)}_{\mathcal{U}\mathcal{L}} & \mathbf{L}_{\mathcal{U}\mathcal{L}} & \mathbf{F}^{(1)}_{\mathcal{U}\mathcal{L}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{6.12}$$

Matrix $\mathbf{Q}[\mathcal{G}_g^- \cup \mathcal{L}, \mathcal{G}^+ \cup \mathcal{U}]$ represents the communication pattern of states in $\mathcal{G}_g^- \cup \mathcal{L}$ to states in $\mathcal{G}^+ \cup \mathcal{U}$ and it is a matrix of zeros except for a finite portion of its g^{th} row that corresponds to entries from g to states in $\mathcal{G}^+ \cup \mathcal{U}^{(1)}$. Let o be a scalar that represents the rate with which state g is left to enter $\mathcal{G}^+ \cup \mathcal{U}$ and can be defined as the sum of all entries on the g^{th} row of $\mathbf{Q}[\mathcal{G}_g^- \cup \mathcal{L}, \mathcal{G}^+ \cup \mathcal{U}]$.

Next, we compute $Norm(\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}])$. From Eq.(6.12), it follows that:

$$\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^-] = \overline{\boldsymbol{\alpha}}^{(0)} \cdot \mathbf{L}^{(0)}_{\mathcal{G}^+\mathcal{G}_g^-} + \sum_{i=1}^{\infty} \overline{\boldsymbol{\alpha}}^{(i)} \cdot \widehat{\mathbf{B}}^{(i)}_{\mathcal{U}\mathcal{G}_g^-}, \tag{6.13}$$

$$\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{L}^{(1)}] = \overline{\boldsymbol{\alpha}}^{(0)} \cdot \widehat{\mathbf{F}}^{(1)}_{\mathcal{G}^+\mathcal{L}} + \overline{\boldsymbol{\alpha}}^{(1)} \cdot \mathbf{L}^{(1)}_{\mathcal{U}\mathcal{L}} + \sum_{i=1}^{\infty} \overline{\boldsymbol{\alpha}}^{(i+1)} \cdot \mathbf{B}^{(i)}_{\mathcal{U}\mathcal{L}}, \tag{6.14}$$

$$\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{L}^{(j)}] = \overline{\boldsymbol{\alpha}}^{(0)} \cdot \widehat{\mathbf{F}}^{(j)}_{\mathcal{G}^+\mathcal{L}} + \sum_{i=1}^{j-1} \overline{\boldsymbol{\alpha}}^{(i)} \cdot \mathbf{F}^{(j-i)}_{\mathcal{U}\mathcal{L}} + \overline{\boldsymbol{\alpha}}^{(j)} \cdot \mathbf{L}_{\mathcal{U}\mathcal{L}} + \sum_{i=j+1}^{\infty} \overline{\boldsymbol{\alpha}}^{(i)} \cdot \mathbf{B}^{(i-j)}_{\mathcal{U}\mathcal{L}} \quad j > 1. \tag{6.15}$$

Let $T = \overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}] \cdot \mathbf{e}$, therefore

$$\begin{aligned} T = \; & \Big(\overline{\boldsymbol{\alpha}}^{(0)} \cdot \Big(\mathbf{L}^{(0)}_{\mathcal{G}^+\mathcal{G}_g^-} + \textstyle\sum_{j=1}^{\infty} \widehat{\mathbf{F}}^{(j)}_{\mathcal{G}^+\mathcal{L}}\Big) + \overline{\boldsymbol{\alpha}}^{(1)} \cdot \Big(\widehat{\mathbf{B}}^{(1)}_{\mathcal{U}\mathcal{G}_g^-} + \mathbf{L}^{(1)}_{\mathcal{U}\mathcal{L}} + \textstyle\sum_{j=1}^{\infty} \mathbf{F}^{(j)}_{\mathcal{U}\mathcal{L}}\Big) \\ & + \textstyle\sum_{i=2}^{\infty} \overline{\boldsymbol{\alpha}}^{(i)} \cdot \Big(\widehat{\mathbf{B}}^{(i)}_{\mathcal{U}\mathcal{G}_g^-} + \textstyle\sum_{j=1}^{i-1} \mathbf{B}^{(j)}_{\mathcal{U}\mathcal{L}} + \mathbf{L}_{\mathcal{U}\mathcal{L}} + \textstyle\sum_{j=1}^{\infty} \mathbf{F}^{(j)}_{\mathcal{U}\mathcal{L}}\Big)\Big) \cdot \mathbf{e}. \end{aligned} \tag{6.16}$$

The term

$$\widehat{\mathbf{B}}^{(i)}_{\mathcal{U}\mathcal{G}_g^-} + \sum_{j=1}^{i-1} \mathbf{B}^{(j)}_{\mathcal{U}\mathcal{L}} + \mathbf{L}_{\mathcal{U}\mathcal{L}} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)}_{\mathcal{U}\mathcal{L}}$$

is bounded by a constant, because of the nature of the infinitesimal generator defined in Eq.(6.1). The $RowSum(\widehat{\mathbf{B}}^{(i)}_{\mathcal{U}\mathcal{G}_g^-} + \sum_{j=1}^{i-1} \mathbf{B}^{(j)}_{\mathcal{U}\mathcal{L}})$ is strictly less than the $RowSum(-\mathbf{L} - \sum_{j=1}^{\infty} \mathbf{F}^{(j)})$, since it is a portion of the positive sum $\widehat{\mathbf{B}}^{(i)} + \mathbf{B}^{(i-1)} + ... + \mathbf{B}^{(1)}$ This results in a finite value of T.

Furthermore, we define

$$\rho = Norm(\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}]) = \frac{\overline{\boldsymbol{\alpha}} \cdot \mathbf{Q}[\mathcal{G}^+ \cup \mathcal{U}, \mathcal{G}_g^- \cup \mathcal{L}]}{T}$$

This vector sums to one (because of the normalization) and can be partitioned according to the levels of $\mathcal{L}$ as:

$$\boldsymbol{\rho} = [\, \boldsymbol{\rho}^{(0)}, \boldsymbol{\rho}^{(1)}, \boldsymbol{\rho}^{(2)}, \ldots].$$

It follows that

$$\overline{\mathbf{L}} = o \cdot \boldsymbol{\rho}^{(0)}$$

and

$$\overline{\mathbf{F}}^{(j)} = o \cdot \boldsymbol{\rho}^{(j)} \quad j \geq 1.$$

Therefore, the sum $\sum_{j=1}^{\infty} \overline{\mathbf{F}}^{(j)} = o \cdot \sum_{j=1}^{\infty} \boldsymbol{\rho}^{(j)}$ converges.

The new process, defined by Eq.(6.11) is ergodic and can be solved using the M/G/1 algorithm outlined in Subsection 3.4. But, in order to apply the algorithm, it is necessary for the sums $\sum_{j=1}^{\infty} \mathbf{F}^{(j)}_{\mathcal{L}\mathcal{L}}$ and $\sum_{j=1}^{\infty} \widehat{\mathbf{F}}^{(0j)}_{\mathcal{G}_g^- \mathcal{L}}$ to converge. $\sum_{j=1}^{\infty} \mathbf{F}^{(j)}_{\mathcal{L}\mathcal{L}}$ converges by definition (see the definition of $\mathbf{Q}$ in Eq.(6.1)). Similarly, $\sum_{j=1}^{\infty} \widehat{\mathbf{F}}^{(0j)}_{\mathcal{G}_g^- \mathcal{L}}$ converges since its component sums are finite. By the definition of $\mathbf{Q}$ in Section 6, the sum $\sum_{j=1}^{\infty} \widehat{\mathbf{F}}^{(j)}_{\mathcal{L}\mathcal{L}}$ is finite.

6.9 Multiple upper-lower classes

In our exposition, we have restricted ourselves to a single pair of upper and lower sets. Here, we present an extension of our upper-lower aggregation algorithm based on recursion that allows us to deal with multiple upper and lower partitions.

The partition is defined by splitting each $\mathcal{S}^{(j)}$, $j \geq 1$, into U upper sets $\mathcal{U}_t^{(j)}$, $1 \leq t \leq U$, and L lower sets $\mathcal{L}_t^{(j)}$, $1 \leq t \leq L$ consistent with a partition of the set of state indices $\{1, \ldots, n\}$ within a level set $\mathcal{S}^{(j)}$ into $U+L$ classes $\mathcal{N}_1, \ldots, \mathcal{N}_U, \mathcal{N}_{U+1}, \ldots, \mathcal{N}_{U+L}$, that is:

$$\forall j \geq 1, \quad \mathcal{U}_t^{(j)} = \left\{ s_i^{(j)} : i \in \mathcal{N}_t \right\} \quad \mathcal{L}_t^{(j)} = \left\{ s_i^{(j)} : i \in \mathcal{N}_{U+t} \right\}.$$

Let $\mathcal{U}_t = \bigcup_{j=1}^{\infty} \mathcal{U}_t^{(j)}$, $\mathcal{U}^{(j)} = \bigcup_{t=1}^{U} \mathcal{U}_t^{(j)}$, and $\mathcal{U} = \bigcup_{t=1}^{U} \mathcal{U}_t$; analogously, $\mathcal{L}_t = \bigcup_{j=1}^{\infty} \mathcal{L}_t^{(j)}$, $\mathcal{L}^{(j)} = \bigcup_{t=1}^{L} \mathcal{L}_t^{(j)}$, and $\mathcal{L} = \bigcup_{t=1}^{L} \mathcal{L}_t$.

Given that L lower sets and U upper sets exist that satisfy the main conditions, i.e., there are no transitions among sets of the same type (except, of course, through the boundary set), there are transitions only from the upper sets to the lower sets, and communication from the lower sets to the upper sets can occur only through the boundary set $\mathcal{S}^{(0)}$, we can apply Algorithm 6.4 in a recursive manner.

(Recursive aggregation of $\mathbf{Q}$ given U upper and L lower classes)

Step 1: **Determine the upper and lower classes in $\mathbf{Q}$.**

Select one lower set $\mathcal{L}_t$, $1 \leq t \leq L$ and redefine $\mathcal{U} = \left(\bigcup_{j=1}^{U} \mathcal{U}_j\right) \cup \left(\bigcup_{j=1, j \neq t}^{L} \mathcal{L}_j\right)$ and $\mathcal{L} = \mathcal{L}_t$, so as to obtain a new partition of the state space that consists of only one upper set $\mathcal{U}$ and one lower set $\mathcal{L}$ such that there is a special "gate" state g in $\mathcal{S}^{(0)}$ such that all paths from $\mathcal{L}$ to $\mathcal{U}$ must first visit g. Sets $\mathcal{G}^+$, $\mathcal{G}_g^+$, $\mathcal{G}^-$, and $\mathcal{G}_g^-$ are defined as in Algorithm 6.4.

If no such partitioning exists, then the algorithm first attempts to solve $\mathbf{Q}$ with a known method, if this is not possible then the algorithm exits. Otherwise, the algorithm continues with step 2.

Step 2: **Generate and solve the "upper CTMC".** Using stochastic complementation identify a new CTMC observing the behavior of the original CTMC only in the states $\mathcal{G}_g^+ \cup \mathcal{U}$. The infinitesimal generator of the new process is $\mathbf{Q}_{\mathcal{G}_g^+ \cup \mathcal{U}}$. Observe that the new "upper" CTMC is composed by the U upper sets and $L - 1$ lower sets of the original partitioning.

Step 2a: If $(L - 1) \geq 1$, then call **Recursive aggregation of $\mathbf{Q}_{\mathcal{G}_g^+ \cup \mathcal{U}}$ given U upper and $L - 1$ lower classes**

Step 2b: If $(L - 1) = 0$, i.e., if the new "upper CTMC" is composed only of the "upper" sets of the original process, then the process defined by $\mathbf{Q}_{\mathcal{G}_g^+ \cup \mathcal{U}}$ is of the GI/M/1-type. This process can be solved using the matrix geometric method, or by applying stochastic complementation to each of its U sets.

Step 3: **Generate and solve the "lower CTMC".** Same as in Algorithm 6.4.

Step 4: **Compute the coupling factors.** Same as in Algorithm 6.4.

Step 5: **Generate the stationary distribution of the original process.** Same as in Algorithm 6.4.

Remarks: We assume that there are U upper sets. Because there is strictly no connection among their infinite parts (except, of course, through the boundary set), applying stochastic complementation to these U sets in the CTMC that results in *Step 2b* is an easy problem. We observe that the single entry condition must be satisfied for each of the stochastic complements to be solved efficiently, that each stochastic complement results in a process of the GI/M/1-type, and that more than one of the new processes may share the same gate state.

To illustrate the idea of the multiple lower and upper sets, we give an example of applying Algorithm 6.9. The CTMC in Figure 6.3(a) has a gate state g_d which we use to construct the pseudo-stochastic complement of the states in $\mathcal{L}_d$. This is *Step 1* of Algorithm 6.9. We proceed with *Step 2* of Algorithm 6.9 and construct the stochastic complement of the rest of the CTMC, which by itself is an independent CTMC as shown in Figure 6.3(b). We continue with *Step 2a* of Algorithm 6.9 and in the new CTMC of Figure 6.3(b) identify a new lower subchain and since *Step 2a* is satisfied we recursively continue with *Step 1* of Algorithm 6.9. The gate state for the second lower subchain is g_c. We built the CTMC of the pseudo-stochastic complement (sets in $\mathcal{L}_c$). We go on with *Step 2* of Algorithm 6.9 and generate the stochastic complement as shown in Figure 6.3(c). Now the *Step 2a* of Algorithm 6.9 is not satisfied and we continue with *Step 2b* by identifying upper sets of the CTMC in Figure 6.3(c). We find the gate states g_a and g_b and built the stochastic complements for the states in $\mathcal{L}_a$ and $\mathcal{L}_b$ respectively as shown in Figure 6.3(d). After solving these two stochastic complements, we can apply the classic aggregation on the sets of the chain in Figure 6.3(c), as shown in Figure 6.3(d). This completes *Step 2b* of Algorithm

6.9 and we continue to *Step 3*, namely solving the lower subsets identified in recursive *Step 1* of Algorithm 6.9. Since we have computed the conditional probabilities of the whole set of upper states, we can proceed with the solution of the pseudo-stochastic complements. Finally, we can apply *Step 4* and *Step 5* of Algorithm 6.9 to complete the solution of the given CTMC.

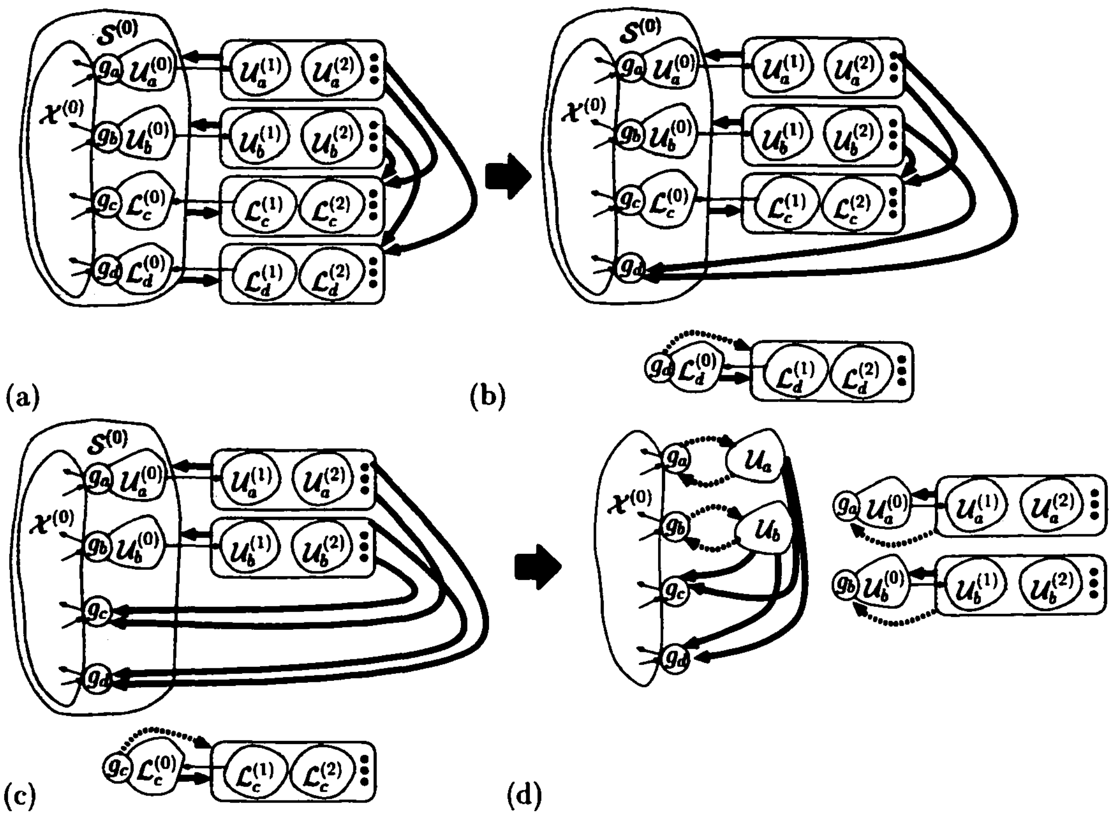
$\mathcal{S}^{(0)}$
$\mathcal{X}^{(0)}$
g_a $\mathcal{U}_a^{(0)}$ $\mathcal{U}_a^{(1)}$ $\mathcal{U}_a^{(2)}$
g_b $\mathcal{U}_b^{(0)}$ $\mathcal{U}_b^{(1)}$ $\mathcal{U}_b^{(2)}$
g_c $\mathcal{L}_c^{(0)}$ $\mathcal{L}_c^{(1)}$ $\mathcal{L}_c^{(2)}$
g_d $\mathcal{L}_d^{(0)}$ $\mathcal{L}_d^{(1)}$ $\mathcal{L}_d^{(2)}$
$\mathcal{U}_a$
$\mathcal{U}_b$
(a)
(b)
(c)
(d)

Figure 6.3: Example of a CTMC with two lower and two upper sets.

We close this section with the following observation. Given the stationary probabilities, we can compute various performance measures of interest such as the expected queue length or its higher moments. The performance measures of interest are computed at *Step 2* and *Step 3* of both Algorithm 6.4 and Algorithm 6.9, where the conditional stationary probabilities of the stochastic complements or the pseudo-stochastic complements are computed and then scaled back according to the coupling factors as defined by *Step 4*.

6.10 Application: parallel scheduling in the presence of failures

We now employ our method to solve a system that can be modeled as a GI/G/1-type CTMC. A popular way to allocate processors among competing applications in a parallel system environment is by space-sharing: processors are partitioned into disjoint sets and each application executes in isolation on one of these sets. Space sharing can be done in a static, adaptive, or dynamic way. If a job requires a fixed number of processors for execution, this requires a static space-sharing policy [99]. Adaptive space-sharing policies [20] have been proposed for jobs that can configure themselves to the number of processors allocated by the scheduler at the beginning of their execution. Dynamic space-sharing policies [55] have been proposed for jobs that are flexible enough to allow the scheduler to reduce or increase the number of processors allocated to them as they run, in response to environment changes. Because of their flexibility, dynamic policies can offer optimal performance; however they are the most difficult to implement because they relocate resources while applications are executing.

Modeling the behavior of scheduling policies in parallel systems often results in CTMCs with matrix geometric form [104]. In these models, neither failures, nor arrivals from multiple exogenous sources are considered. To illustrate the applicability of our methodology, we present instead a CTMC that models the behavior of a scheduling policy in a cluster environment subject to software and hardware failures. Our system is a cluster composed of two sub-clusters connected via a high speed medium (e.g., Gigabit Ethernet), while the nodes in each sub-cluster are connected via a lower speed switch (e.g., ATM switch). For simplicity, we present a small system composed of two sub-clusters only, where a limited number of possible partitions is allowed, but our methodology readily applies to larger systems with multiple partitions. The arrival process is Poisson with parameter λ but each arrival may be a bulk of arbitrary size governed by a geometric distribution with parameter ρ. For clarity's sake, we only draw arcs corresponding to bulks of size one and two only, labeled with the rates $\lambda_1 = \lambda\rho$ and $\lambda_2 = \lambda(1-\rho)\rho$, respectively.

The system employs a space-sharing policy that allows up to two parallel jobs to run simultaneously. A parallel job may execute across the whole cluster (i.e., on both sub-clusters) or occupy only one sub-cluster: the service time is exponentially distributed with rate μ_2 in the former case, or μ_1 in the latter. The policy is as follows. Upon arrival

- while there are no jobs in the system:
  - if the arrival is of a single job, that is, the bulk size equals one, the whole

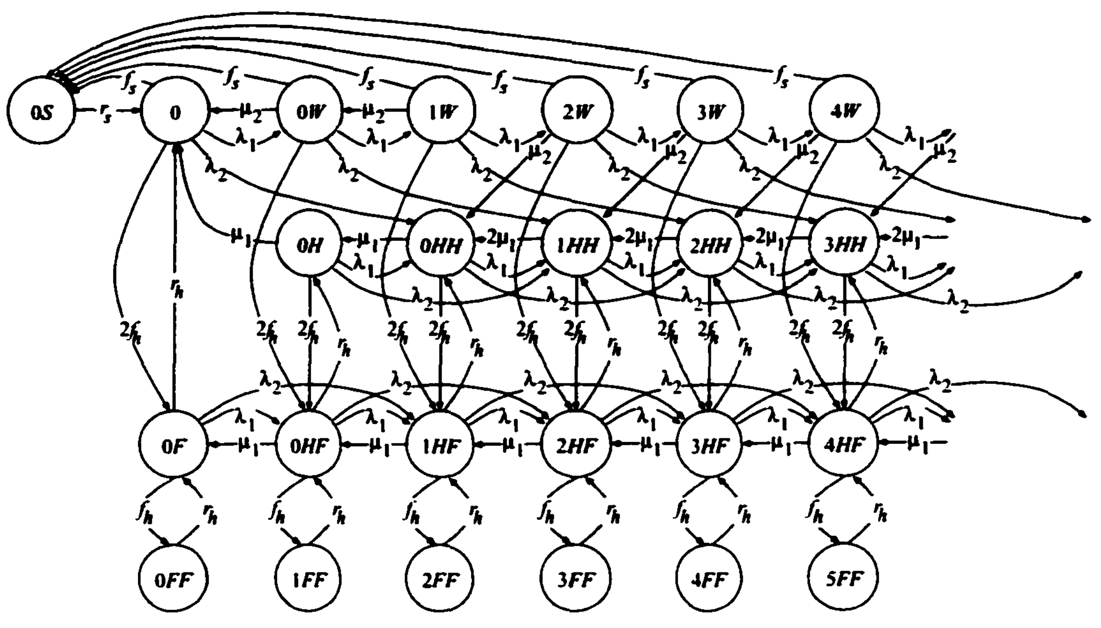

Figure 6.4: The CTMC for our performability study

cluster is assigned to that job,

- otherwise, if multiple jobs arrives simultaneously, that is, the bulk size is greater than one, two jobs are scheduled, one on each sub-cluster, while the remaining jobs in the bulk, if any are queued;

- while there are already jobs in the system and one job running using the whole cluster:
  - if the bulk size equals one the job is simply queued,
  - otherwise, if the bulk size is greater than one, the arriving jobs are enqueued, the current job is stopped and restarted on a single sub-cluster, and one of the queued jobs is also started, on the other sub-cluster;

- while there are jobs in the system already and two jobs are running, each on one sub-cluster:
  - we simply enqueue arriving jobs, regardless of the bulk size.

Upon a completion (departure)

- of a job that was using the entire system:
  - if there is only one job waiting, it is assigned the entire system.
  - if there are multiple jobs waiting, two of them are assigned a sub-cluster each;
- of a job that was using only one sub-cluster:
  - one of the jobs in the queue, if any, is assigned the sub-cluster just released (if there are no jobs waiting, the other job running, if any, is *not* reassigned to the entire system).

The rationale behind these decisions is that, under bursty conditions, we would like to reach our goal of having smaller partitions quickly, even at the cost of killing a running job and rescheduling it on a smaller partition [90].

We consider the performance of our scheduling policy under the following failure scenarios. Each of the two subclusters can experience a local hardware failure independently of each other; in this case only the affected sub-cluster must be brought down for repair, and the system can still accept arrivals. In addition, when a parallel job is assigned the entire cluster, it makes use of software whose execution can cause the entire system to crash. After such a failure, each sub-cluster must be brought down for reboot; consequently, the system's queue is flushed and no arrivals are accepted until repairs are completed. We assume that all event durations are exponentially distributed. The rate for a hardware failure is f_h, for a software failure f_s, for a repair after a hardware failure is r_h, and for a repair (reboot) after a software failure is r_s.

State	Description
0	Empty system, no failures
0S	Empty system, rebooting after a software failure
kW	$k \geq 0$ jobs in the queue, one job executing on the whole system
$0H$	0 jobs in the queue, one sub-cluster idle, one other running a job
kHH	$k \geq 0$ jobs in the queue, each sub-cluster running a job
$0F$	0 jobs in the queue, one sub-cluster idle, the other failed
kHF	$k \geq 0$ jobs in the queue, one sub-cluster running a job, the other failed
kFF	$k \geq 0$ jobs in the queue, both sub-clusters failed

Table 6.2: State description of the CTMC.

Figure 6.4 depicts a CTMC modeling the behavior of our system and the Table 6.10 describes the meaning of the system states. The infinitesimal generator of this CTMC has a GI/G/1 structure consistent with the requirements of our solution algorithm, since we can immediately specify state 0 as the gate state, states kW plus $0S$ as the upper states, and the states $0H, kHH, 0F, kHF$ and kFF as the lower states. Note that the model assumes that hardware failures do not occur during reboot; if this were not true we could add a transition from $0S$ to $0F$ with rate $2f_h$, and the structure required by our approach would still be present, since there can be transitions from $\mathcal{U}_g^{(0)}$ (i.e., $0S$) to any state in $\mathcal{L}_g$.

The numerical values used for our numerical parameters are: $f_h = 10^{-6}, \ldots, 10^{-3}$, $f_s = 10 \cdot f_h$, $r_h = 10^{-3}$ or 10^{-1}, $r_s = 10 \cdot r_h$, $\mu_1 = 3.0$ or 6.0, $\mu_2 = 1.8\mu_1$, $\lambda = 10^{-5}, \ldots, 5$, and $\rho = 0.8$.

Figure 6.5(a) and Figure 6.5(b) show that long term availability of the system, that is, its ability of accepting arrivals, computed as the stationary probability of being in any of the states except $0S$ and kFF, for $k \geq 0$, for a given choice of r_h and two choices of μ_1, and for a given choice of μ_1 and two choices of r_h, respectively, as a function of f_h and λ (note that the arrival rate λ) does affect the availability, since it affects the probability of the system being in the kW states, hence of software failures. Figure 6.5(c) and Figure 6.5(d) are analogous, except that they focus on the probability on not being unavailable due to a software failure, that is, they plot the complement of being in state $0S$. Figure 6.5(e) and Figure 6.5(f) focus instead on the system power, defined as the ratio of the system throughput over the average job response time, also as a function of f_h and λ and for various choices of r_h and m_1. It is apparent that there is a significant correlation between the above parameters and the workload that the system can handle. Note that, in all plots, the missing data points correspond to parameter combinations for which the system becomes unstable.

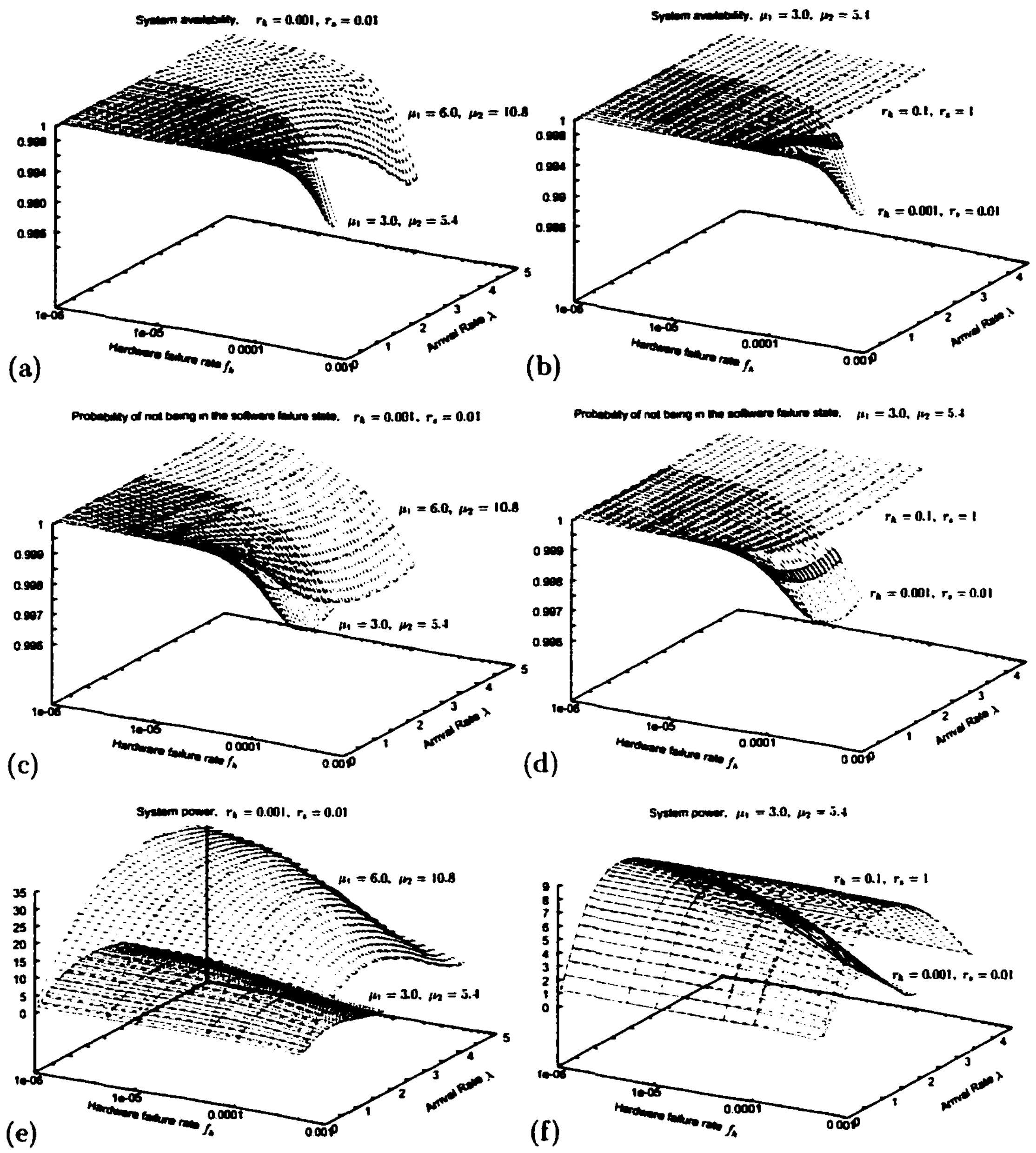

Figure 6.5: Performance results.

6.11 Chapter summary

In this chapter we presented an aggregation-based algorithm for the exact analysis of a class of GI/G/1-type Markov chains with repetitive structure. Such chains appear when modeling systems with failures and repairs that also accept arrivals from multiple exogenous sources. The algorithm also applies to systems that are purely of the M/G/1-type, or GI/M/1-type, or their intersection, that is quasi-birth-death processes. Consequently, it extends both the applicability and efficiency of well-known techniques for the solution of M/G/1-type and GI/M/1-type processes including the matrix geometric and matrix analytic methods through an intelligent *partitioning* of the repetitive portion of the state space into subsets, according to its connectivity. This partitioning allows us to define smaller CTMCs related to the original CTMC, then solve their stochastic complements using established techniques.

Our algorithm does not apply to all GI/G/1-type CTMCs. Here, we describe the exact conditions that must be satisfied to allow application of our algorithm but we do not present a partitioning algorithm of the process state space required to apply our method. This is a graph partitioning problem and is subject of future work. We note, however, that the nature of the system under examination may guide us into easily identifying the possible partitioning of the state space.

Chapter 7

Load Balancing on Clustered Web Servers

Popular Internet sites are experiencing continuous increase in the number of users and amount of requests they serve [17]. In such conditions, maintaining high user-perceived performance is challenging, but important for the commercial success of the site. A common solution is implementation of Web sites as distributed systems with several mirrored Web servers, transparent to the user, known as clustered Web servers. They are popular because they allow Web sites to scale as customer population grows and also ensure Web site availability and reliability.

In a distributed environment, the management of resources, i.e., the load balancing policy, has ample importance, because it significantly affects user perceived performance. The load balancing policy in a cluster of Web servers assigns the stream of incoming requests to the servers of the cluster, striving for fast processing time per request and maintaining equally utilized servers. Since the load balancing policy deals with the stream of requests, its performance highly depends on the characteristics of the cluster workload.

Throughout this dissertation, we stressed that in a Web server the request interarrival process exhibits long-range dependence, and the request processing time is highly variable. Detailed analysis of the effects of these characteristics on the performance of the load balancing policy, allows us to develop policies that are aware of the cluster workload, and improve user perceived performance. In the following sections, we describe how we characterize the service process in a Web server by PH distribution and model it as a queueing system for performance analysis purposes. Such analysis guides us on the design of new load balancing policies that maintain low request slowdown even under dynamically changing workload characteristics.

This chapter is organized as follows. In Section 7.1, we give an overview of different Web cluster architectures. Section 7.2 outlines load balancing policies associated with clustered Web servers. Section 7.3 outlines the analytic models that we propose to analyze the performance of load balancing policies in Web server clusters. In Section 7.4, we describe a sized-based load balancing policy for clustered Web servers, and analyze its performance using analytic models. In Section 7.5, we propose EQUILOAD that is a refinement of the sized-based policy. In Section 7.6, we propose ADAPTLOAD, a load balancing policy that continuously adapts its parameters to the characteristics of the cluster workload. We conclude the chapter with a summary of the presented results.

7.1 Clustered Web servers architectures

Web server clusters are classified based on the assignment of the incoming requests to the various servers of the cluster [17] as follows:

- **Client-based approach.** In this approach, the client host itself routes the request to one of the servers in the cluster. This can be done by the *Web-browser* or by the *client-side proxy-server.* The client communicates with the servers in order to decide which server will eventually serve the request. The range of load balancing policies that can be implemented is limited, but this approach is scalable and provides high availability.

- **DNS-based approach.** The DNS server for the domain of the clustered server is the responsible device to distribute the incoming requests among servers of the cluster. This approach allows various load balancing policies to be implemented in the cluster. Nevertheless, DNS has a *limited* control on the requests reaching the Web cluster. After the client has resolved the IP address of the server in the cluster, the intermediate name servers between the client and the cluster cache the logical name for the IP address and save it for a specific amount of time (Time-To-Live, TTL). Any request that goes through any of these intermediate servers, does not reach the DNS server of the cluster. Two algorithms exist that implement the DNS approach depending on the value of TTL, constant TTL and dynamic TTL algorithms.

- **Dispatcher-based approach** (at network level). This is an alternative to DNS-based approach, that provides full control on client requests and masks the request routing among multiple servers. The cluster has only *one virtual IP address*, the IP address of the dispatcher. The dispatcher identifies the servers through unique *private* IP addresses, which are used by the dispatcher for scheduling purposes. The routing of request can be done either via packet rewriting or with HTTP redirection. The dispatcher controls all the incoming requests, so its involvement on the load balancing algorithm is kept at a minimum. A simple high-level design of a dispatcher-based architecture is depicted in Figure 7.1.

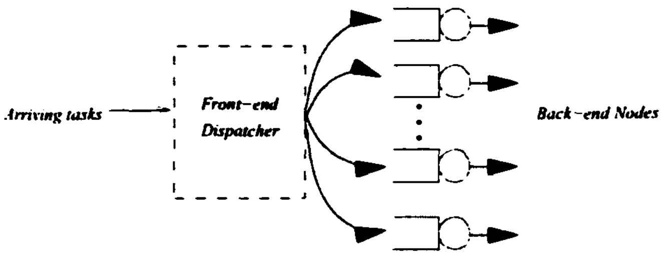

Figure 7.1: Model of a distributed server.

- **Server-based approach.** This approach uses a two-level dispatching mechanism. Client requests are initially assigned by the DNS server to a server in the cluster but each server may reassign a received request to any other server in the cluster. This approach solves some of the problems that arise with the DNS approach. The server based approach may use either packet rewriting or HTTP redirection to reroute requests among servers.

For the remaining of this chapter, we focus on dispatcher-based and server-based Web server cluster architectures.

7.2 Load balancing policies on clustered Web servers

One of the most compelling problems that arises on distributed Web server clusters is the selection of an efficient load balancing policy. The load balancing policy should strive to evenly utilize servers in the cluster, and achieve minimum response time for each request. The simplest load balancing policies are *random* and *round robin* policies. The former assigns an incoming request to a uniformly randomly selected server, while the latter assigns the requests to the servers of the cluster in a "cyclic" fashion. Round robin is widely used because it is easy to implement and implies only a minimum overhead. A common variation of the round robin policy is the *weighted round robin* policy [25, 41, 73]. In the weighted round robin policy, the incoming requests are distributed among the servers in a round robin fashion, weighted by some measure of the load on each of the servers. Policies like *join shortest queue* and *shortest remaining processing time* are cases of the weighted round robin policy, where the front-end dispatcher requires detailed information about the load and operation of each server in order to assign incoming requests to servers of the cluster.

In addition to the traditional load balancing policies, more sophisticated policies exist. They take into consideration the special characteristics of both the architecture and the workload of clustered Web servers. A priori knowledge of the characteristics of the incoming stream of requests, such as the size of the requested files, their location (i.e., in disk or memory) is incorporated in recently proposed load balancing policies for clustered Web servers [17, 5, 36, 73, 19, 112]. Such load balancing policies are called "content-aware" policies.

Among the content-aware policies, we distinguish the *locality aware* load balancing policies [73, 19, 112]. Locality-aware policies improve cluster's performance by distributing incoming requests based on the cache content of each server, since a request is served much faster if it fetched from the cache of the Web server than from its local disk.

Other policies incorporate knowledge of the variability in the service process into the load balancing policy [36]. SITA-E (Size Interval Task Assignment with Equal Load) assigns the incoming requests into servers based on the request size, assuming that the requested file sizes follow a bounded Pareto distribution.

The characteristics of the server-based architecture are used to propose additional load balancing policies by redirecting the requests among servers of the cluster according to a "domain-based" partition of the servers [18, 2]. This partition changes dynamically as a function of changes in the cluster service demands. These policies are further extended by allowing quality-of-service requirements to define the set of servers that serves an incoming request [115].

7.3 World Cup 1998 workload

The workload, we use to analyze load balancing policies in clustered Web servers, consist of the server logs of the 1998 World Soccer Cup Web site[1]. The World Cup site server was composed of 30 low-latency platforms distributed across four geographically distributed locations. Client requests were dispatched to a location via a Cisco Distributed Director, and each location was responsible for load balancing of incoming requests among its available servers. The traces provide information about each request received by the site during 92 days, from April 26, 1998 to July 26, 1998. Trace data were collected for each day during the total period of time that the Web server was operational.

The traces provide information about each request received by each server. For each request the following information is recorded: the IP address of the client issuing the request, the date and time of the request, the URL requested, the HTTP response status code, and the content length (in bytes) of the transferred document. Trace data were collected for each day during the total period of time that the Web server was operational. Since the focus of this work is on load balancing, irrespective of possible caching policies at the server, we only extracted the content length of the transfered document from each trace record assuming that the service time of each request is a linear function of the size of the requested document. Analysis of the unique file size distribution across all 92 days indicated that the service process in

[1]Available at `http://ita.ee.lbl.gov/`.

the Web server is heavy-tailed. This trend persists also if the empirical distribution of the sizes of requests files is analyzed on a day by day basis. For a detailed analysis of the World Cup workload see [3].

7.3.1 Fitting request sizes into a distribution

Analysis of Internet traffic at different levels of the communication infrastructure shows that many processes in Internet-related systems are highly variable and best characterized by heavy-tailed distributions [7, 8, 4, 3, 30]. Detailed analysis of server logs [7, 3] shows that file-size requests are best described by *hybrid* distributions, where the body is best described by *lognormal distribution* and the tail by a *power-tailed* distribution [3].

To check for the heavy-tail property, we used Boston University's **aest** tool that verifies and estimates the heavy-tail portion of a distribution [28][2]. Using the scaling estimator methodology, the tool helps identify the portion of the data set that exhibits power-tailed behavior by demonstrating graphically the tail of the distribution where the heavy-tailed behavior is present. The selection of the point where the power-tailed behavior starts is significant because it affects the computation of the parameters of the distribution. Figure 7.2 shows the results of the scaling analysis for the service process of a representative day of the dataset, i.e., day 80. Considering the tail portion of the plots, for requests larger than 1 MByte, we see that they are close

[2]http://www.cs.bu.edu/faculty/crovella/aest.html.

to linear, suggesting that the heavy-tailed portion of the dataset begins at around 1 MByte. Based on this observation, we conclude that the empirical distribution is best approximated by a hybrid model that combines a lognormal distribution for the body of the data with a power-tailed distribution for its tail [3]. After identifying the

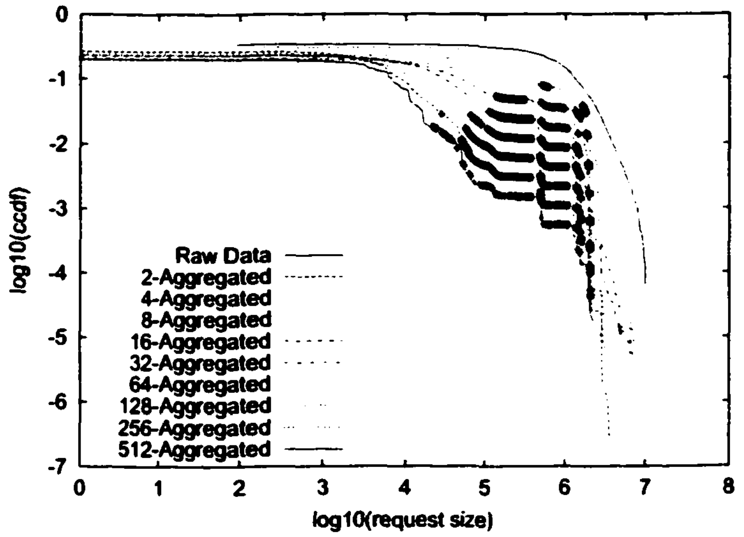

Figure 7.2: Tail characterization for day 80. The various curves in the figure show the ccdf of the dataset on a log-log scale for successive aggregations of the dataset by factors of two. The figure illustrates that the shape of the tail (i.e., for size $> 10^6$) is close to linear and suggests the parameter for its power-tailed distribution. The '+' signs on the plot indicate the points used to compute the α in the distribution. The elimination of points for each successive aggregation indicates the presence of a long tail.

two portions of the trace, we need to compute the parameters of each of its portions. The body of the distribution is considered lognormal with PDF:

$$f(x) = \frac{1}{bx\sqrt{2\pi}} \exp\left(\frac{-(\ln x - a)^2}{2b^2}\right).$$

We compute $b > 0$ (i.e., the shape parameter), and $a \in (-\infty, \infty)$ (i.e., the scale parameter) using the maximum likelihood estimators [48]:

$$\hat{a} = \frac{\sum_{i=1}^{n} \ln X_i}{n}, \quad \hat{b} = \left[\frac{\sum_{i=1}^{n} (\ln X_i - \hat{a})^2}{n}\right]^{\frac{1}{2}}.$$

where X_i for $1 \leq i \leq n$ are the sample data. The trace is heavy-tailed with tail index α if its CDF is:

$$P[X > x] \sim x^{-\alpha}, \quad x \to \infty, \quad 0 < \alpha < 2,$$

where X is the random variable describing the request size. In our study, we compute α via the **aest** tool.

Once the data is approximated by a hybrid distribution, we apply the FW[3] algorithm [30] for approximating a heavy-tailed distribution with a hyperexponential one. Since the body and the tail of the hybrid distribution are heavy-tailed, yet described by different distribution functions, we apply the algorithm to each component separately and finally combine both fittings into a single hyperexponential, weighting each part accordingly. The weights of the two hyperexponential distributions, corresponding to the lognormal and the power-tailed portions of the original data, are given by the probability that a request is for a file with size less or equal to, or greater than, 1 Mbyte, respectively. These weights are computed from the empirical data. The final result is a hyperexponential distribution fitting for the entire data set.

[3]We choose FW algorithm to fit the hybrid distribution into a hyperexponential because here we are dealing with distribution functions and FW performs well, i.e., it is fast and accurate, when fitting heavy-tailed distribution functions.

7.3.2 Fittings using FW algorithm

We use the FW algorithm to fit the data representing the request sizes from two representative days of the World Cup 98 site, day 57 and day 80, into hyperexponential distributions. Figures 7.3 and 7.4 illustrate the CDFs of the actual data, their fittings into a hybrid distributions (lognormal and power-tail), and the fittings of the hybrid distribution into a hyperexponential one. We observe that the resulting hyperexponential distribution closely matches the behavior of the original data.

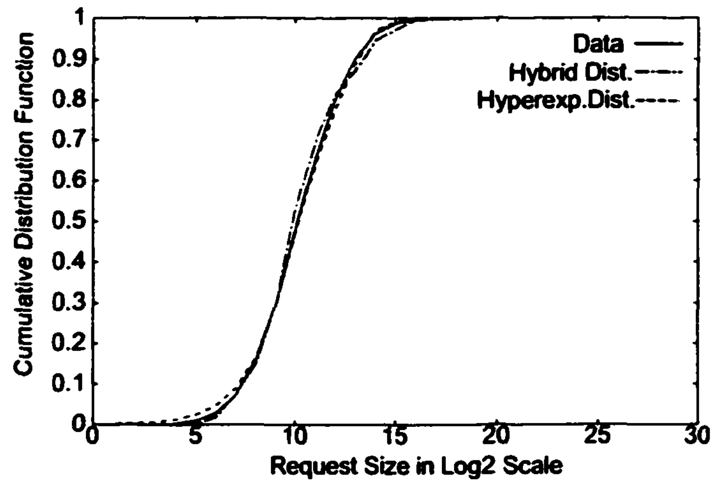

Figure 7.3: Fitted data of day 57

Table 7.1 illustrates the parameters of the lognormal and the power-tailed portions of the distribution for each day. For both days, we see that the bulk of the data lies in the lognormal portion of the data set, while only a very small part (albeit with very large file sizes) lies in the power-tailed part. Table 7.1 also shows the parameters that the FW algorithm suggests for the fitting of the above distributions

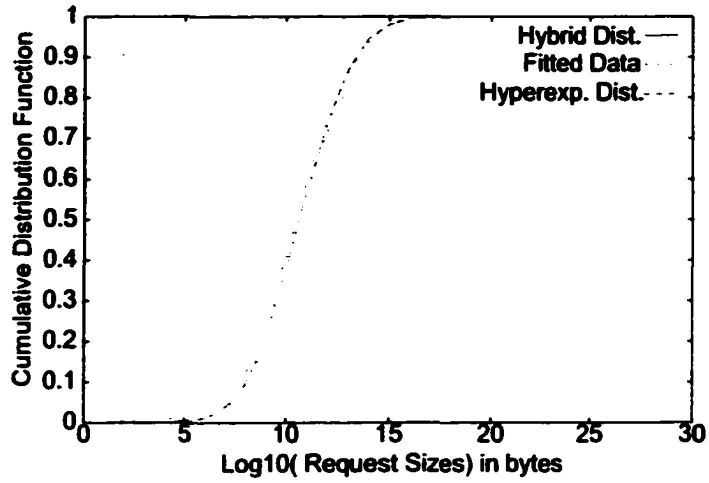

Figure 7.4: Fitted data of day 80

into hyperexponentials. In both cases, a total of seven exponential phases, four for the lognormal portion and three for the power-tailed portion, are sufficient to achieve an excellent approximation of the original data.

Data from Day 57			Data from Day 80		
Lognormal(a, b) $a = 7.033358$ $b = 1.509296$	Power(α) $\alpha = 0.82$	Weight for Lognormal 0.99935	Lognormal(a, b) $a = 7.43343$ $b = 1.42824$	Power(α) $\alpha = 0.89$	Weight for Lognormal 0.999977
Parameters of the $H_7(\mu_i, \beta_i : 1 \leq i \leq 7)$ fitting			Parameters of the $H_7(\mu_i, \beta_i : 1 \leq i \leq 7)$ fitting		
μ_i	β_i		μ_i	β_i	
0.000000008469532	0.000000000438327		0.000000013708911	0.000000000190479	
0.000000106031775	0.000000002331229		0.000000215667700	0.000000001569731	
0.000011317265198	0.000649997230444		0.000043293323498	0.000569998239790	
0.000001510047810	0.000014700266903		0.000005312880260	0.000744062014933	
0.000018914686309	0.014450795383216		0.000029646684825	0.039321723492673	
0.000190007539767	0.443701366678877		0.000150999863822	0.367916087712180	
0.001235615667834	0.541183137671004		0.000870285230628	0.591448126780214	

Table 7.1: Workload parameters for day 57 and day 80

Because our objective is to analyze the performance of load balancing policies under variable service process, we also considered synthetically-generated data sets that exhibits different variabilities. We base our synthetic service process generation on the properties of day 57. Since the power-tailed portion of each of the selected days is very small, we turn our attention to the lognormal portion which also exhibits heavy-tailed behavior. By changing simultaneously both the *a* and *b* parameters of a lognormal distribution, we change both the scale and shape of the distribution so as to vary the variance of the distribution and at the same type keep the mean of the distribution constant (equal to the mean of day 57, i.e., 3629 Bytes). This way, we can examine the sensitivity of our load balancing policies to the variability of the service process. Figure 7.5 illustrates how the CDF changes when we change the

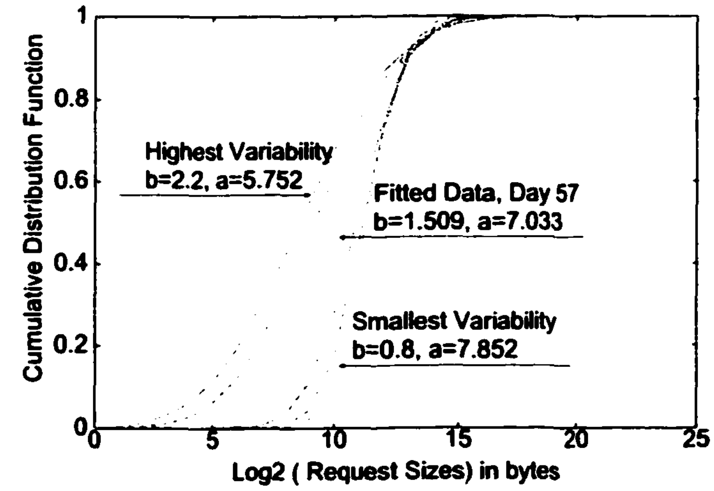

Figure 7.5: Shape of the CDF when changing the variability of the lognormal portion of the data set

variability in the data set (the fitting technique resulted in three, four, or five stages for the lognormal fitting, thus a total of six, seven, or eight stages were used to fit the mixture of the lognormal and power-tailed distribution). We evaluate the sensitivity of the policy as a function of variability in the service process in Section 7.4.

7.4 Sized-based load balancing policy

We consider the following model of a distributed server environment. We assume a fixed number c of back-end servers with the same processing power, each serving requests in first-come-first-serve order. We further assume that each server has an unbounded queue. Requests arrive to the dispatcher according to a Poisson process. The dispatcher is responsible for distributing the jobs among the various back-end servers according to a scheduling policy. We also assume that the dispatcher can derive the request duration (the size of the file) from the name of the file requested. We consider the following two load balancing policies (in neither case the dispatcher uses feedback from the individual back-end servers to better balance the load, measured in "bytes to be transferred", among them):

Random: The dispatcher assigns the incoming request to a randomly selected server, with probability $1/c$.

Size-based: The dispatcher assigns requests to servers according to their size. This policy is motivated by the desire to separate large from small requests, to avoid the significant slowdowns that small requests would experience when queued behind large ones. A policy based on the same principle has been examined in [36] and compared very favorably to a dynamic policy where the dispatcher assigns requests to servers according to the server's load at the time of request arrival.

We first consider the performance of the random policy under Poisson arrivals and service process that is driven by the entries of the synthetically-generated requested file sizes described in Subsection 7.3.2. Recall that one of the curves in Figure 7.5 corresponds to the actual stream of requested file sizes corresponding to day 57 of the World Cup server logs. In the analysis of this section, we assume that the overall arrival rate to the dispatcher is λ, and that there are eight back-end servers. Since the processing time for each request is linear to request size, we approximate the service process at the Web server by the hyperexponential distribution of requested file sizes and model its performance using an M/Hr/1 queue. In the case of the random policy, each Web server in the cluster is model by the same M/Hr/1 queue with arrival rate $\lambda/8$. An M/Hr/1 queue results in a QBD which we analyze using ETAQA-QBD (see Section 5.3).

In most of our experiments, slowdown is the metric of interest that we choose for evaluation of cluster performance. Average request slowdown, the ratio of response time to service time for a job, is a fundamental measure of responsiveness in Web servers, since users are more willing to wait for "large" requests than for "small" ones. We start our evaluation by analyzing the performance of the random load balancing policy. Our objective is to be able to understand how the variable service process affects the performance of the random policy and identify possible improvements.

The average request slowdown for the random policy is illustrated in Figure 7.6(a). Although the system saturates at the same value of λ regardless of the variability in the service process (recall that all data sets have the same mean request size), the average request slowdown differs dramatically from service process to service process, especially in the range of medium-to-high system utilization. Figure 7.6(b) illustrates the average queue length at each server as a function of the variability in the service process for various arrival rates[4]. The figure further confirms that the higher the variability in the service process, the more dramatic the average queue build-up is.

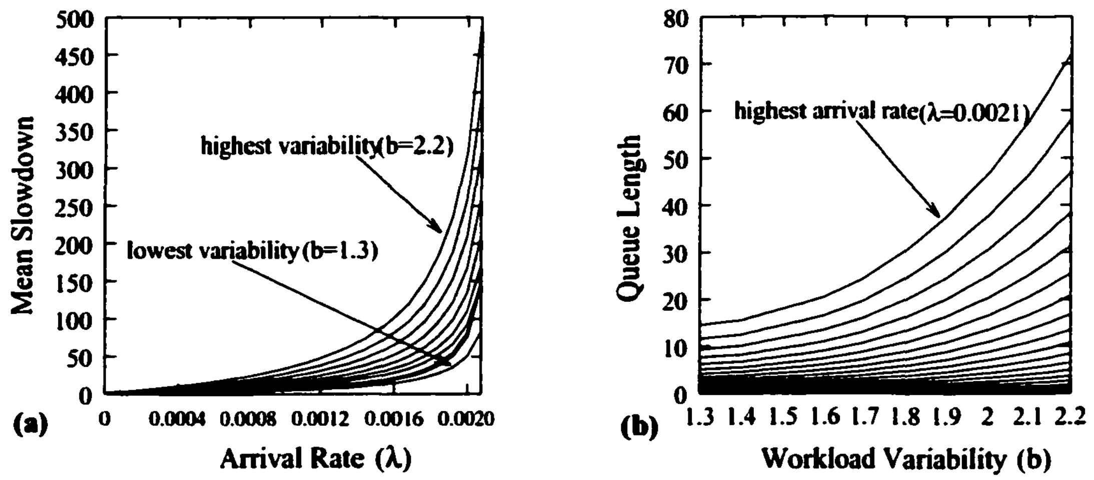

Figure 7.6: Average request slowdown as a function of the overall request arrival rate λ for high-to-low variability in their request service times **(a)**, and average queue length as a function of the variability in the service process for various arrival rates **(b)**.

[4]For presentation clarity, the x-axis of Figure 7.6(b) shows only the value of the b parameter of the lognormal distribution for the corresponding data set, but we remind the reader that a different value of b implies also a different value of a, to keep the same mean request size across all data sets.

To further understand the behavior of the system under the random balancing policy, we look closer at the range of request sizes that contribute to the queue build-up and consequently to the performance degradation. Our analytical model allows us to further explore the system behavior by analyzing how the queue length builds up. Figure 7.7 sketches the CTMC that models a back-end server, an M/Hr_7/1 server (for presentation clarity, not all arcs are illustrated in the picture, but the reader can visualize the shape of the Markov chain and, most importantly, identify the parts of the CTMC that correspond to the power-law portion of the workload and the lognormal portion of the workload).

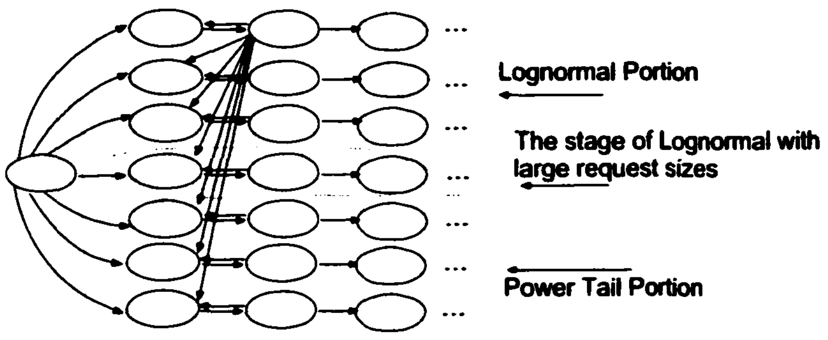

Figure 7.7: The CTMC that models one host.

The model of Figure 7.7 is analyzed using ETAQA-QBD which allows the *exact* computation of the stationary system queue length. Figure 7.8 illustrates the contribution to the overall queue length from the instances when short requests get stuck behind long ones, i.e., behind requests for files larger than 1MB (files that are part of the power-tailed portion of the distribution) and 100 KBytes (the tail part of the lognormal distribution together with the power-tailed part of the distribution). Since the FW algorithm "splits" the distribution (each portion corresponding to one phase of the hyperexponential distribution), it is possible to calculate approximately the contribution to the queue length of instances of queue built-ups because short requests wait for longer ones to be served. Figure 7.8(a) illustrates that, at medium-to-high load, the queue built-up due to requests from the power-tailed portion begin served is about 20% of the overall system queue length (even if the frequency of these requests is almost negligible). This percentage is much larger at smaller arrival rates.

We also note that if the lognormal portion has low variability, the queue build-up is dominated by instances of short requests waiting behind long ones, which we call *power-tailed* queue. This can be explained by examining the contribution to the queue build-up by the tail of the lognormal distribution. Figure 7.8(b) shows that the requests from the tail of the lognormal distribution play an important role on the performance (requests for files larger than 100 KBytes dominate the queue across the entire range of arrival rates) and illustrates that for higher variabilities in the service process, the system queue length due to large yet rare requests is significant.

These last observations suggest that it may be appropriate to assign requests to specific servers according to their sizes. We conjecture that by reserving servers for scheduling requests of similar sizes, we ensure that no severe imbalances in the utilization of each of the servers occur.

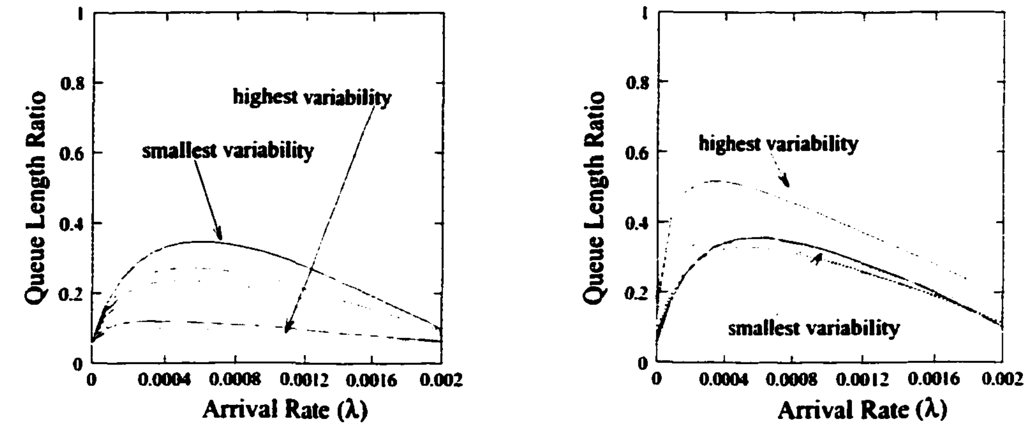

Figure 7.8: Ratio of queue built up behind requests of files greater than 1MByte **(a)**, and of files greater than 100 KBytes **(b)**.

The fitting provided by the FW algorithm for the requested files sizes provides a hyperexponential distribution with a special property: each exponential phase corresponds to a certain range of request (file) sizes. Thus, we use the hyperexponential distribution to make an educated guess on distributing the incoming requests across the back-end servers, ensuring that the variance of the service time distribution of the requests served by each server is kept as low as possible. Figure 7.9 illustrates this size-based policy.

By applying the size-based policy to our data sets with the stream of requested files sizes, we notice that a single server suffices to serve the power-tailed portion and the tail of the lognormal portions of the request file sizes distribution. The body of the lognormal portion must instead be served by the remaining seven servers. Figure 7.10 illustrates the average queue length of the hosts using either the size-based or the random policy for two fixed arrival rates, $\lambda = 0.0012$ and $\lambda = 0.0016$,

1. Compute the expected service time S_i for each phase of the hyper-exponential distribution, weighted by its probability β_i:
$S_i = \frac{\beta_i}{\mu_i}, \quad 1 \leq i \leq k$

2. Normalize S_i to compute each stage's contribution to the overall expected mean service time of the distribution:
$\hat{S}_i = \frac{S_i}{\sum_{i=1}^{k} S_i}, \quad 1 \leq i \leq k$

3. If c servers are available, then phase i should be served by
$c_i = \hat{S}_i \cdot c$ servers
(the specific server for the request is chosen randomly among the c_i servers)

4. Treat heavy-tail differently from the body of the distribution:

   a. $\forall c_i < 1, \quad 1 \leq i \leq k$, (i.e., for stages corresponding to the heavy tail) such that $\sum c_i < 1.5$, are to be served the *same* single server.

   b. $\forall c_i \geq 1, \quad 1 \leq i \leq k$, (i.e., for stages corresponding to the body), assign $\lfloor c_i + 0.5 \rfloor$ servers and schedule jobs within these servers using the random policy. Attention should be paid so as to ensure that the total server assignment across all stages of the hyper-exponential does not exceed c.

Figure 7.9: Our size-based scheduling policy.

representing medium and high load, respectively.

In contrast to the random policy, the average queue length with the size-based policy does not increase as the variability in the service process increases. This indicates that the size-based policy, being aware of the heavy-tailed behavior in the service process, sustains better the variability effects in the Web server performance. Figure 7.10 shows similar results with respect to the expected request slowdown. We conclude that taking into consideration the properties of the service process in the load balancing policy, as we do in the sized-based policy, insures better overall performance of the Web server cluster.

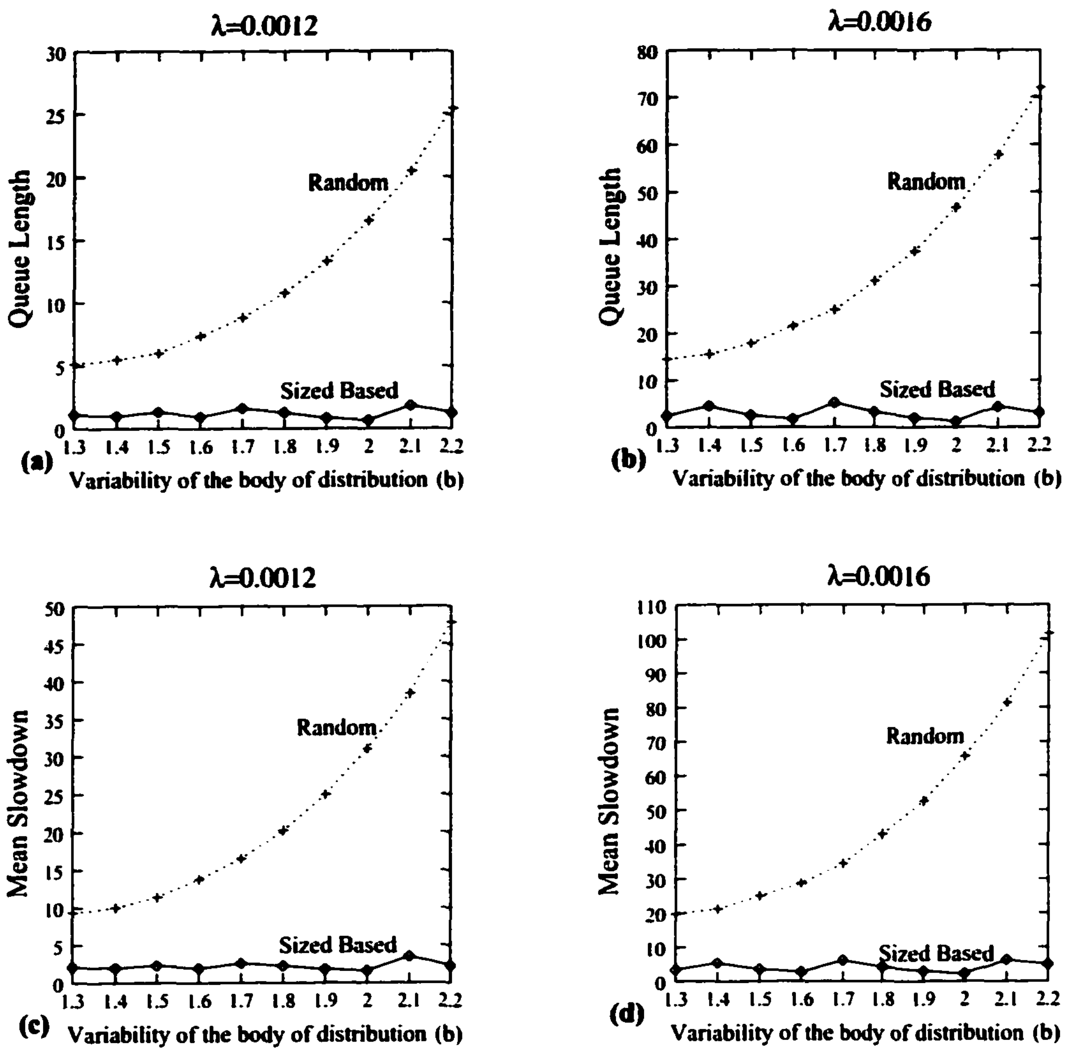

Figure 7.10: Policy comparisons as a function of the workload variability.

7.5 EquiLoad

The results presented in Section 7.4 show that classifying the requests that arrive into a cluster of Web servers according to their processing time (assuming linear dependence with the size of the requested file) improves cluster performance and handles well the variability in the service process.

Nevertheless, the policy has two drawbacks: (a) the front-end dispatcher does not always know the file sizes of each request in advance and (b) several servers may serve jobs within the same size-class, which further complicates the policy by requiring efficient scheduling within this subcluster of servers.

We address these two problems and propose a load balancing policy for clustered Web servers, which we call EQUILOAD. This policy requires partitioning the possible request sizes into N intervals, $[s_0 \equiv 0, s_1), [s_1, s_2), \ldots, [s_{N-1}, s_N \equiv \infty)$, so that server i is responsible for processing requests of size between s_{i-1} and s_i. In practice, the size corresponding to an incoming URL request might not be available to the front-end dispatcher, but this problem can be solved using a two-stage allocation policy. First, the dispatcher assigns each incoming request very quickly to one of the N back-end servers using a simple policy such as uniform random (or round-robin, which is even easier to implement in practice). Then, when server i receives a request from the dispatcher, it looks up its size s and, if $s_{i-1} \leq s < s_i$, it puts the request in its queue, otherwise it redirects it to the server j satisfying $s_{j-1} \leq s < s_j$ (of course any request that server i receives from another server is instead enqueued immediately, since it is guaranteed to be in the correct size range). This policy looks to be similar with the server-based architecture described in Subsection 7.1, but we propose to have a front-end dispatcher not a DNS server that handles the incoming requests. Such policies that are based on redirecting requests among servers in a cluster have been implemented [2, 18].

Letting the back-end servers reallocate requests among themselves is sensible, since the size information is certainly available to them. The potential advantages of such policy are clear:

- Partitioning the stream of requested files into N class sizes and assigning each to a different server maximizes the homogeneity of request sizes within each queue, thus improves the overall response time.
- The dispatcher does not need to be aware of the sizes of the incoming requests, nor of the load of each back-end server.
- Requests must be reallocated at most once; indeed, if the dispatcher uses a simple random or round-robin allocation, the probability that a request must be reallocated is exactly $\frac{N-1}{N}$.
- Except for the small reallocation overhead, no server's work is wasted.
- As there is no overlapping on size ranges of the N classes, there is no sub-clustering of the servers.
- The cache behavior is guaranteed to be close to optimal[5], since requests for the same file are assigned to the same server.

but so are the challenges:

[5]All our experiments are driven by the servers logs at the World CUP 1998 Web site. Using this workload, we always achieved with EQUILOAD almost optimal caching behavior. In the remainder of this chapter, whenever we claim close to optimal behavior for EQUILOAD, we assume that the Web cluster operates under similar workload as the one measured at the World CUP 1998 Web site.

- The value of the $N-1$ size boundaries $s_1, s_2, \ldots, s_{N-1}$ is critical, since it affects the load seen by each server; the mechanism used to choose these boundaries must achieve the desired performance goals, in our case, maintaining a equal expected load, measured in "bytes to be transferred", at each server.
- As the workload changes over time, we must dynamically readjust these boundaries, but the cost for doing so must be small.

For the first challenge, the objective is to provide each back-end server with (approximately) the same "load" by choosing the intervals $[s_{i-1}, s_i)$ so that the requests routed to server i contribute a fraction $1/N$ of the mean $\overline{S}$ of the distribution size. In other words, we seek to determine $s_1, s_2, \ldots, s_{N-1}$ such that, for $1 \le i \le N$,

$$\int_{s_{i-1}}^{s_i} x \cdot dF(x) \approx \frac{1}{N} \int_0^{\infty} x \cdot dF(x) = \frac{\overline{S}}{N}.$$

where $F(x)$ is the CDF of the requested file sizes. Given a trace of R requests and their sizes, the s_i boundaries can be determined using the algorithm outlined in Figure 7.11. EQUILOAD builds a discrete data histogram (DDH) [48] encoding the empirical size distribution of the trace requests. We can think of the DDH as a vector of (b, c) pairs, where b is a particular size of requests encountered in the trace (in bytes), c counts the number of times requests of this size appear in the trace, and the vector entries are sorted by increasing values of b. From the DDH, we can easily compute the expected request size, $\overline{S}$. We scan the DDH and accumulate the sizes and their frequencies, recording the points $s_1, s_2, \ldots, s_{N-1}$ at which we reach a fraction $\frac{1}{N}$, $\frac{2}{N}$, $\ldots$, $\frac{N-1}{N}$ of $R \cdot \overline{S}$. We address the second challenge in Section 7.6.

```
1.  compute the DDH of the sizes of the $R$ requests for $F$ different files:
    $\{(b_f, c_f) : 1 \le f \le F\}$;
        $b_f$ indicates size in bytes
        $c_f$ indicates size frequency
2.  compute the total requested bytes: $B \leftarrow \sum_{f=1}^{F} c_f b_f$
3.  initialize the accumulated size and DDH index: $A \leftarrow 0$ and $f \leftarrow 0$
4.  for $i = 1$ to $N - 1$ compute $s_1, s_2, \ldots, s_{N-1}$ by scanning the DDH:
    a.  while $A < B \cdot i/N$ do
        I.   increment $A$ by the contribution of entry $f$ of the DDH: $A \leftarrow A + c_f b_f$
        II.  move to the next entry of the DDH: $f \leftarrow f + 1$
    b.  set the $i^{\text{th}}$ boundary so that $\sum_{f=1}^{s_i} c_f b_f = B \cdot i/N$: $s_i \leftarrow f$
```

Figure 7.11: Setting the boundaries $s_1, s_2, \ldots, s_{N-1}$ with EQUILOAD.

7.5.1 Performance of EquiLoad

EQUILOAD needs to identify the boundaries of the request sizes assigned to each server, and it does this by partitioning the set of possible sizes, from 0 to ∞, into N disjoint intervals, such that the same expected load is seen by all servers. Since we intend to analyze EQUILOAD using analytical models, we fit the measured stream of requested file sizes into PH distributions. In particular, we require that the data falling in a given interval, i.e., the load in a single server, be fitted in a PH distribution so as to be able to analyze the performance of individual servers using matrix-analytic techniques. The D&C MM fitting method, proposed in Section 4.3, is an appropriate fitting technique, since it divides the data set into N partitions with equal expected value and fits each of them into a PH distribution.

The PH fitting for the request sizes allows us to model the Web server as an M/PH/1 queue. These type of queueing systems can be analyzed using ETAQA (proposed in Chapter 5). In particular, a general M/PH/1 queue is a QBD process and we use ETAQA-QBD to solve it. Hence, we evaluate the performance of the entire cluster by analyzing the performance of individual servers. In the case of the random policy. we adapt the M/PH/1 queueing model of the entire cluster to each server by correcting the arrival rate to reflect only $1/N$ of the overall cluster arrival rate. Performance of each server under EQUILOAD is analyzed using the respective M/Hr/1 or M/Hypo/1 queueing model for that particular server.

We first compare EQUILOAD performance with that of the random policy. We based our analysis in the synthetically-generated stream of requested file sizes described in Figure 7.5. Recall that one of the curves in Figure 7.5 corresponds to the measured stream of requested file sizes during day 57 of the World Cup 1998 logs. In all experiments of this subsection, we assume that the requests arrive to the cluster following a Poisson distribution with rate λ and that there are four back-end servers. Thus, the arrival rate to each individual server under the random policy is $\lambda/4$. The results presented here are obtained as solutions of queueing models using ETAQA-QBD. We note that whenever possible, i.e., for the case of day 57, the analytic performance measures are validated via trace driven simulation and are in excellent agreement.

We compare the expected queue build-up of the Random and EQUILOAD policies as a function of variability in the service process. We concentrate on performance figures for medium and high arrival intensity into the system (see Figure 7.12). Contrary to the Random policy, the expected queue length with EQUILOAD does not increase as a function of the variability in the service process. EQUILOAD exhibits a remarkable ability to select the size range boundaries for each server so as to keep the expected queue build-up to a minimum, thus offers a simple and inexpensive solution that is significantly better than the Random policy.

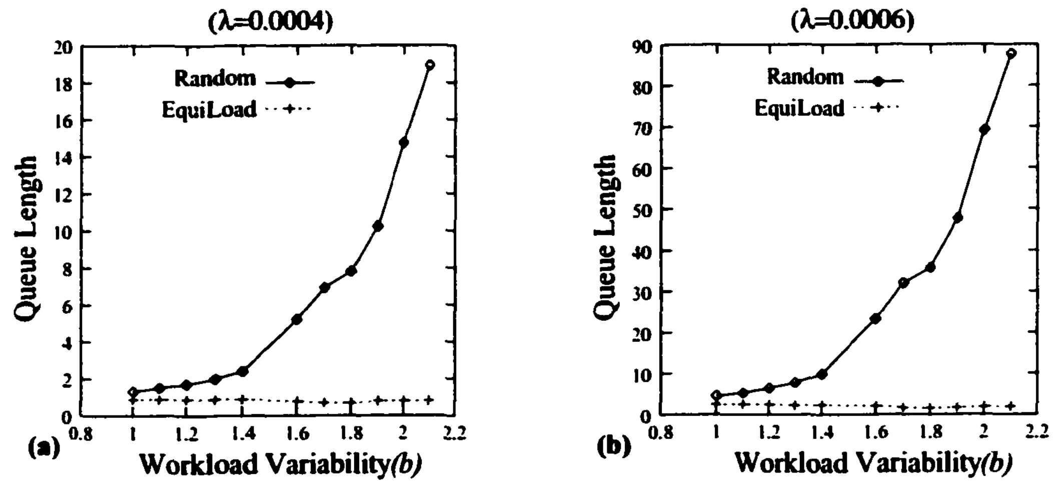

Figure 7.12: Average queue length of Random and EQUILOAD policies as a function of the variability in the service process for medium arrival rate ($\lambda = 0.0004$) and high arrival rate ($\lambda = 0.0006$).

To assess the performance of EQUILOAD with respect to the variability in the service process and system load, we compare it with the SRPT and JSQ load balancing policies. SRTP assigns each incoming request to the server that is expected to finish first the requests already assigned to it. The dispatcher makes this assignment decision based on detailed information about the completion time for each request in all servers of the cluster. The performance of SRPT under Web-related service processes is analyzed in detail in [37], where it is shown that SRPT outperforms typical scheduling policies currently in use for Web servers, such as processor sharing. JSQ assigns each incoming request to the server with the shortest queue. In this policy, the front-end dispatcher must know only the size of the waiting queue for each server in the cluster. In a distributed environment, this policy handles the system load reasonably well [113, 114].

We compare EQUILOAD with SRPT and JSQ via trace-driven simulations. Figure 7.13 illustrates the expected queue length in each server of the cluster as function of variability in the service process, for a medium ($\lambda = 0.0004$) or heavy ($\lambda = 0.0006$) system load. The expected task slowdown (not shown) closely follows the observed behavior for system queue length. SRPT handles well both the variability of the service process and the load of the cluster. JSQ performs slightly worse than SRPT but it does not require as much information to be available to the front-end dispatcher. When the variability in the service process or the system load are not very high, both SRPT and JSQ perform slightly better than EQUILOAD (the "jagged" EQUILOAD curve is an artifact of the fittings). In more critical cases for the operation of the cluster, such as higher variability in the service process or higher system load, EQUILOAD performs better than SRPT and JSQ (note the scale difference of the y-axis between Figures 7.13(a) and 7.13(b)).

Observing that the performance of EQUILOAD is comparable to that of SRPT and JSQ in non-critical cases, but substantially better in critical ones. Furthermore, EQUILOAD does not require the front-end dispatcher to have any additional knowledge about the processing and the load of each server in the cluster, while both SRPT and JSQ require the front-end dispatcher to know in detail (SRPT even in more detail than JSQ) the status of operation for each server in the cluster.

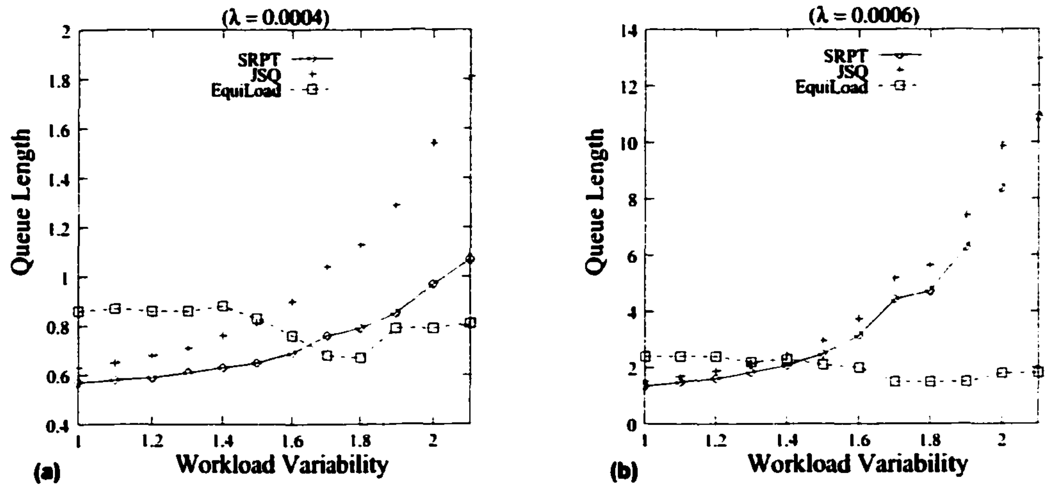

Figure 7.13: Average queue length of SRPT, JSQ and EQUILOAD policies as a function of variability in the service process for medium arrival rate ($\lambda = 0.0004$) and high arrival rate ($\lambda = 0.0006$).

We conclude that EQUILOAD does not perform as well as SRPT or JSQ for lightly loaded system, or low variability in the service process, because in these situations the servers assigned to "short" jobs are more loaded than those assigned to "long" jobs: with low variability in the service process or system load, "long" requests are infrequent.

7.5.2 Locality awareness of EquiLoad

A fundamental property of EQUILOAD is that files within a specific size range are sent to the same server. This increases the chances that a given request can be satisfied from the local cache, resulting in a service time that is considerably less than when the requested file is not currently in the cache but only in the disk. In the previous performance analysis, we assumed that files are always fetched from the disk, i.e., that the service time of each request is linearly related to the actual file size. However, the effect of local caching in load balancing of clustered web servers significantly affects performance. LARD is a "locality-aware" scheduling policy that takes into consideration the cache contents of the back-end servers [73] and offers significant performance advantages versus non locality-aware policies.

By its nature, we expect the cache behavior of the back-end servers with EQUILOAD to be close to optimal, since requests for the same file are assigned to the same server. To examine EQUILOAD's performance with respect to caching, we adjust our simulation implementation and ensure that the service time for each request is computed according to the storage location (cache or disk) from where the requested file is fetched. We assume that the service time is two orders of magnitude smaller if the file is fetched from cache instead of local disk. We stress that even in this version of EQUILOAD the dispatcher does not require any information about the status of the back-end servers.

For comparison, we simulate a LARD-like load balancing policy where the dispatcher knows the cache contents of each server in the cluster and assigns each incoming request to the least-loaded server that has already the requested file in its local cache, if there is one, otherwise to the least-loaded server. To avoid the possibility of extreme load unbalances, if all servers that have the requested file in their local cache are heavily loaded compared to those that don't have it, the dispatcher assigns the request to the server with the lightest load (more precisely, this happens if the difference between the loads of the least-loaded server among those with the request in the cache and the least-loaded server among those without the request in the cache exceeds a given threshold T). In our experiments, we use different values of the threshold T, to investigate the values yielding the best performance.

The simulations run on a real trace, the day 57 of the World Soccer Cup. Figures 7.14(a), and Figure 7.14(b), illustrate the average system queue length and the average system cache hit ratio respectively, as function of the arrival rate[6]. We note that EQUILOAD handles the increasing load in the system very well and has the highest and constant cache hit ratio. EQUILOAD outperforms the LARD-like policy for all threshold values T, and all system loads. The LARD-like policy, although not as good as EQUILOAD, also performs well. The higher the values of threshold T, the better the performance of the system for high system loads, while for low system loads the smaller the threshold values the better the performance. However, the LARD policy

[6]Note that the system can sustain higher arrival intensities comparing to the results of previous subsections. This is a direct outcome of the fact that the service rates are much higher (by two orders of magnitude) if the file resides in cache.

does not sustain a high cache-hit ratio as the system load increases.

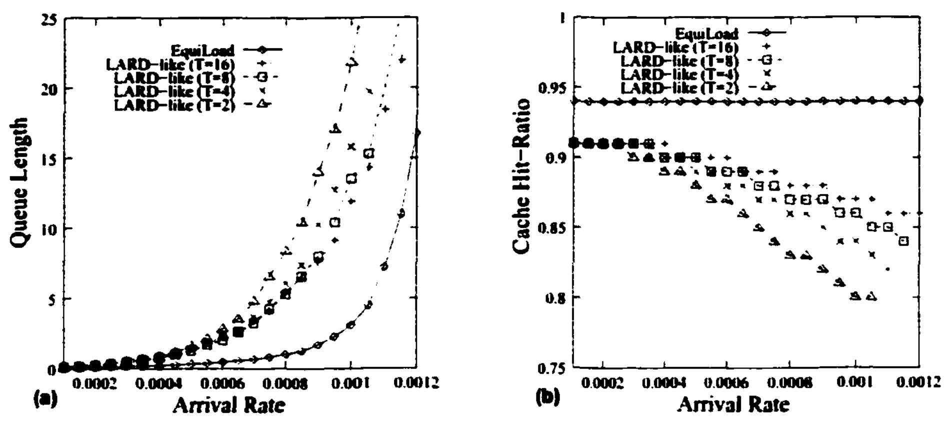

Figure 7.14: Average queue length and cache hit-ratio of LARD-like and EQUILOAD policies as a function of the arrival rate.

7.5.3 Behavior of EquiLoad policy in a dynamic environment

We have established the fact that EQUILOAD is an effective policy for clustered Web servers. It is important to note that the effectiveness of EQUILOAD is related to the selection of the intervals $[s_{i-1}, s_i)$, i.e., the range of request sizes allocated to server i. Our policy determines these intervals so that the requests routed to each of the N servers contribute a fraction $1/N$ of the mean $\overline{S}$ of the distribution size. The policy effectiveness is ultimately tied to determining appropriately these intervals. Given that there is clear evidence that the workload changes not just from day to day but could also change from hour to hour, it becomes increasingly important to be able to dynamically adjust the size intervals accepted by each server.

To assess the importance of selecting the appropriate endpoints for each of the s_i intervals, we selected six traces from the World Cup 1998 web site. each trace

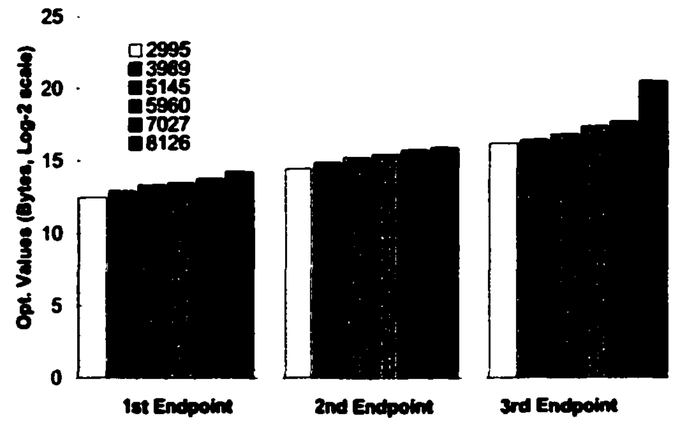

Figure 7.15: Optimal boundaries for the various workloads.

representing a distinct day. The means of the distribution of each trace are equal to 2,995, 3,989, 5,145, 5,960, 7,027, and 8,126 bytes for days a, b, c, d, e, and f, respectively. Figure 7.15 illustrates the values of the s_1, s_2, and s_3 size boundaries of a four servers cluster operating under the requests of the six traces.

Figure 7.16 shows the system expected queue length as a function of the arrival intensity for days b, c, d, and f. The solid curve in each graph (labeled "optimal") corresponds to the queue length curve when the optimal partitioning for the particular trace data is used as defined by the EQUILOAD algorithm. Figures 7.16(a) and (b) show that if the computed intervals correspond to a past portion of the trace that is close enough to the current portion then the expected behavior is close to the optimal. If instead the computed intervals correspond to a trace portion with very different behavior than the current portion (e.g., day f in comparison to days b and

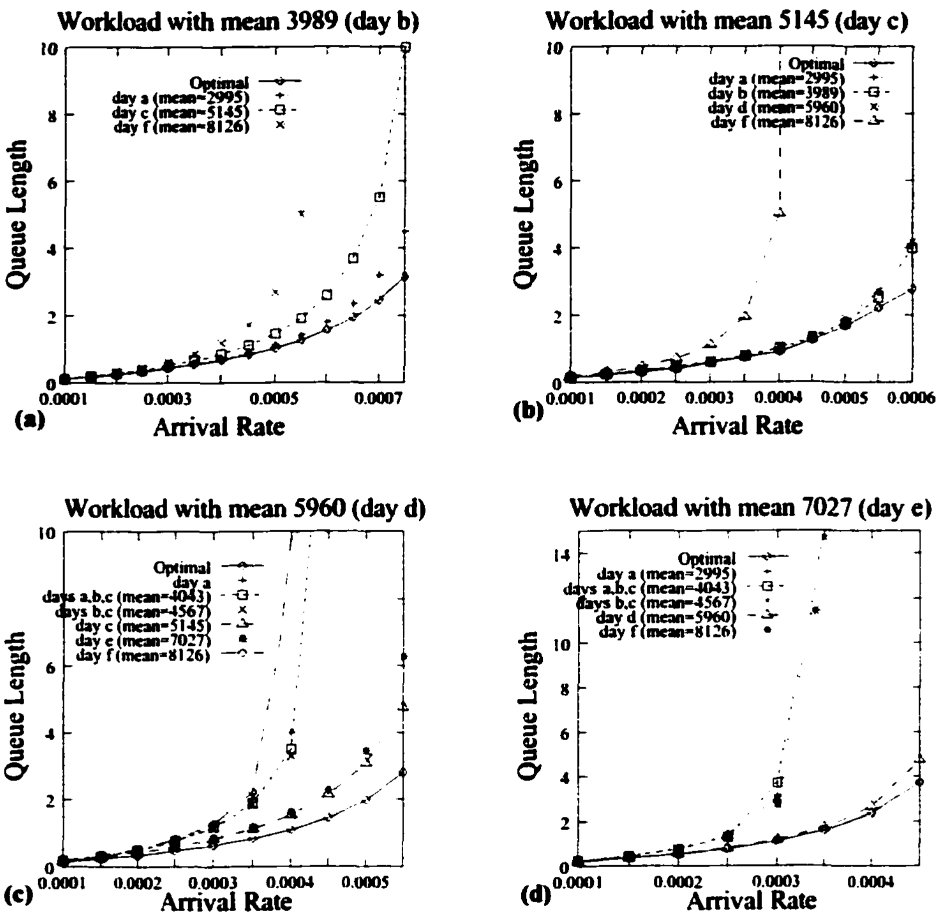

Figure 7.16: Average queue lengths for various size boundaries for days *b*, *c*, *d*, and *e* .

c), then the system reaches saturation much faster. Since day f has a much higher mean than both days b and c, its intervals are shifted toward the right and favor load unbalancing in each host's queue.

Figures 7.16(c) concentrates on the queue build-up when the stream of requested file sizes is from day d and the policy parameters are computed using the portion of the trace that corresponds to days a, c, e, f, the trace that results when merging the trace portions of days a, b, and c, as well as the trace the that results when merging the trace portions of days b and c. Under the assumption that day a strictly precedes days b, that day b strictly precedes day c, and that day c strictly precedes day d, we see that computing the policy parameters from the very immediate past (i.e., only day's c portion) may be more beneficial than using a more extended past history. The same behavior is further confirmed in Figure 7.16(d). Overall, Figure 7.16 highlights the need for a dynamic on-line algorithm that intelligently adjust its parameters based on the characteristics of the current workload in the cluster.

7.6 AdaptLoad

In Section 7.5, we proposed EQUILOAD, a scheduling policy that balances the load in a cluster of Web servers by using the size of each incoming request for server assignment. EQUILOAD advocates dedicating servers to requests of similar sizes. Exact *a priori* knowledge of the request size distribution is an essential requirement for EQUILOAD performance, and its main weakness as well, since such knowledge may not always be available. For example, special events may drastically alter the relative popularity of the Web server documents or result in new documents being offered. In this section, we leverage EQUILOAD by providing an on-line mechanism that *continuously adapt* the policy parameters to changes in the incoming request pattern. The load balancing policy we propose, called ADAPTLOAD, dynamically re-adjusts its parameters based on system monitoring and on-the-fly characterization of the incoming workload.

We evaluate ADAPTLOAD via trace-driven simulation using real trace data from the 1998 World Cup Soccer Web site. The selected trace data contains significant fluctuations in the workload arrival pattern, thus is a good means to evaluate the performance of a policy that strives to adapt to changing workload behavior. Our simulations indicate that knowledge of a finite portion of the past workload can be used as a good indicator of future behavior. Using a geometrically discounted history of the workload starting from the immediate past, we tune the values of the $N - 1$ interval boundaries and show that an intelligent adaptation of these boundaries is not only feasible but it can also provide excellent performance.

7.6.1 Transient characteristics of World Cup workload

As ADAPTLOAD adjusts its parameters exclusively according to the distribution of the incoming workload, detailed knowledge of the workload statistical characteristics is essential to policy effectiveness. Thus, we first present the salient characteristics of the workload used in our analysis, the traces of the 1998 World Soccer Cup (described in Section 7.3).

Figure 7.17 focuses on the arrival and service characteristics of eight "busy" days with high total traffic intensity for the 1998 World Cup, namely from June 23 1998 to June 30, 1998. In all four plots, data is collected at 5-minute intervals. Figure 7.17(a) illustrates the arrival rate into the system as a function of time and shows that there is a clear periodic pattern on a per-day basis. Six out of the eight days show a sharp increase (two spikes) in the arrival intensity during the evening hours, a direct outcome of posting the results of distinct soccer games. This effect is observed also during the other two days, June 27 and June 28, although not as markedly as with the rest of the selected days. Figure 7.17(b) illustrates the total volume of transferred data for the same 8-day period.

To investigate the correlation between the arrival intensity and the size of each request, we plot the average request size experienced by the site during the same period, and observe that there is negative correlation between the arrival intensity and the request size: peaks in Figures 7.17(a) and (b) correspond to dips in Figure 7.17(c), suggesting that the most popular files during the busy evening periods have a small size. Figure 7.17(d) plots the coefficient of variation of the request size, and further confirms that the requests during the high arrival intensity periods are more uniform in size than the rest of the day. However, coefficients of variation as high as ten indicate the presence of highly variable, i.e., long-tailed, behavior.

This analysis illustrates the difficulty of policy parameterization. The policy parameters need to swiftly adapt to changes in the request distribution, which can vary dramatically from an hour to the next within the course of a day. Consequently, these parameters must be tuned carefully, especially when high arrival intensities are expected.

7.6.2 AdaptLoad: the algorithm

EquiLoad is based on the observation that directing tasks of similar size to the

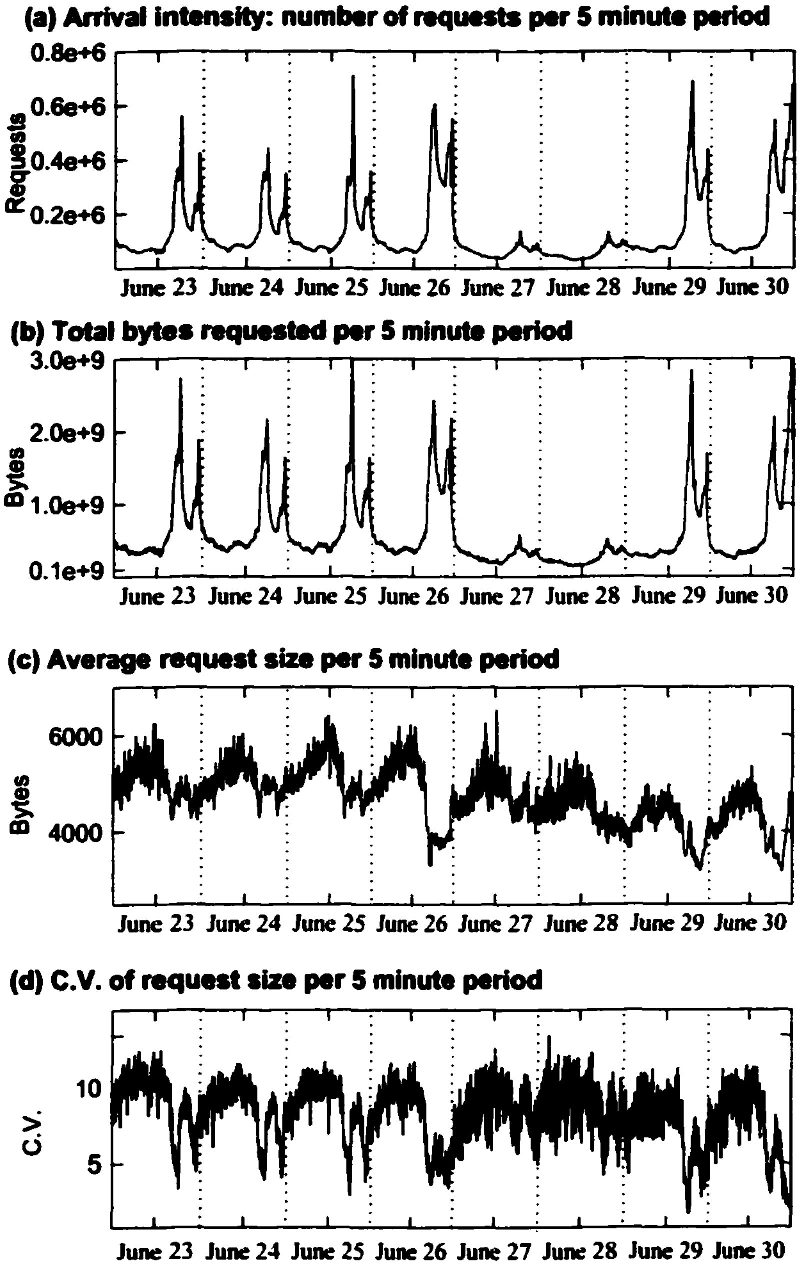

Figure 7.17: Arrival and service characteristics of a "busy" week of the 1998 World Cup trace.

same server reduces the request slowdown in a Web server. In a cluster of N web servers, EQUILOAD requires partitioning the possible request sizes into N intervals.

$[s_0 \equiv 0, s_1)$, $[s_1, s_2)$, up to $[s_{N-1}, s_N \equiv \infty)$, so that server i, $1 \leq i \leq N$, is responsible for satisfying the requests with size falling in the i^{th} interval. EQUILOAD relies on accurate computation of its parameters and, in Subsection 7.5.3, we demonstrated that EQUILOAD performance degrades if its parameters do not reflect the current workload characteristics. In Subsection 7.6.1, we discussed how the workload can change its characteristics across not only successive days but also within a single day. Therefore, a dynamic adjustment of the s_i boundaries is imperative for high performance.

One simple solution to the above problem is to use the system history, more precisely the last K requests seen by the system, to build the DDH needed to determine the boundaries for the allocation of the next K requests (recall that EQUILOAD determines the policy parameters using the DDH of the request sizes). K must be chosen wisely: it should be neither too small (since it must ensure that the computed DDH is statistically significant) or too large (since it must allow adapting to workload fluctuations). In Section 7.6.3, we provide a refined algorithm and use the requests history in a geometrically-discounted fashion as a better prediction technique.

An additional algorithmic modification is necessary to ensure good performance, given that the boundaries are computed using an empirical distribution. If a significant portion of the requests consists of a few popular files, it may not be possible to select N *distinct* boundaries and still ensure that each interval $[s_{i-1}, s_i)$ corresponds to a fraction $\frac{1}{N}$ of the load.

To solve this problem, we introduce "fuzzy" boundaries. Formally, we associate a probability p_i to every "fuzzy" boundary point s_i, for $i = 1, \cdots, N-1$, expressing the portion of the requests for files of size s_i that is to be served by server i. The remaining portion $1 - p_i$ of requests for this file size is served by server $i + 1$, or even additional servers (for the 1998 World Cup data, sometimes we had to extend a fuzzy boundary beyond two servers to accommodate a very popular file).

Figure 7.18 illustrates AdaptLoad which, unlike the algorithm EquiLoad of Figure 7.11, uses past requests information to determine (fuzzy) boundary points for the future. Also, since AdaptLoad is an online algorithm, it must manipulate DDHs efficiently (in linear time). Thus, instead of storing a DDH with a vector of size equal to the number of files, we use a vector with a constant number F of *bins*, so that, for $1 \le f \le F$, bin f accumulates the total number of bytes t_f due to requests for files with size between C^{f-1} and C^f, in the (portion of the) trace being considered[7]. Thus a DDH is now expressed simply as $\{(f, t_f) : 1 \le f \le F\}$. Accordingly, the (possibly fuzzy) boundaries are expressed in terms of bin indices, not actual file sizes.

7.6.3 Performance analysis of AdaptLoad

This subsection presents a detailed performance analysis of AdaptLoad via trace-driven simulation. We selected traces for two consecutive days, June 26 and 27, as

[7] C is some real constant greater than one. Using a value close to one results in a fine DDH representation, but also in a larger value for F, since C^F must exceed the size of the largest file that may be requested.

```
1. set fuzzy boundaries $(s_1, 1), (s_2, 1), \ldots, (s_{N-1}, 1)$ to "reasonable guesses"
2. initialize:
   a.  request counter: $R \leftarrow 0$
   b.  total requested bytes: $B \leftarrow 0$
   c.  DDH: $\forall f, 1 \leq f \leq F, t_f \leftarrow 0$
3. while $R < K$
   a.  get new request, let its size be $s$, and increment $R$
   b.  assign request to a server based on $s$ and $(s_1, p_1), (s_2, p_2), \ldots, (s_{N-1}, p_{N-1})$
   c.  insert request size into the DDH bin $f$ such that $C^{f-1} \leq s < C^f$: $t_f \leftarrow t_f + s$
   d.  increment $B$ by the new request size: $B \leftarrow B + s$
4. initialize the server index $i \leftarrow 1$ and the accumulated weight $A \leftarrow 0$
5. for $f = 1$ to $F$ scanning the DDH bins to update $(s_1, p_1), (s_2, p_2), \ldots, (s_{N-1}, p_{N-1})$:
   a.  increment accumulated weight $A$ with the weight of the $f^{th}$ bin: $A \leftarrow A + t_f$
   b.  while $A > B/N$
       I.    set boundary $s_i$ for server $i$ to the current $f$: $s_i \leftarrow f$
       II.   decrement $A$ by $B/N$: $A \leftarrow A - B/N$
       III.  set fraction $p_i$ for server $i$ to be: $p_i \leftarrow 1 - A/t_f$
       IV.   increment server index $i$: $i \leftarrow i + 1$
6. go to 2. and process the next batch of $K$ requests
```

Figure 7.18: Setting the fuzzy boundaries $(s_1, p_1), (s_2, p_2), \ldots, (s_{N-1}, p_{N-1})$ with ADAPTLOAD.

representative (see Figure 7.17). During these two days, 52 and 18 million requests were served, respectively. From each trace record we select two values, the request arrival time and the size of the requested file (in bytes).

As described in the previous section, ADAPTLOAD balances the load on the back-end servers using its knowledge of the past request sizes distribution. Specifically, the algorithm of Figure 7.18 schedules the i^{th} batch of K requests according to boundaries computed using the $(i-1)^{th}$ batch of K requests[8]. As expected, the performance of the

[8]For the first batch, previous history is not available. In our simulation we simply discard its corresponding performance data, so we use the first K requests just to compute the first DDH. In a practical implementation, instead, we would simply guess boundaries to be used for the first batch.

policy is sensitive to the value of K. For the World Cup 1998 data, we experimented with several values for K and $K = 50{,}000$ appears to be a good choice. Indeed, values between 25,000 and 75,000 results in almost indistinguishable performance, while values in the hundred of thousands are poor due to the high variability over time in the arrival process. In Section 7.6.6, we elaborate on alternative ways to use previous requests to predict the future ones.

We compare AdaptLoad against the *Join Shortest Weighted Queue* (JSWQ) policy, where the length of each queue in the system is weighted by the size of queued requests[9]. We focus on the following questions:

Can AdaptLoad *respond quickly to transient overload?* To answer this question, we report performance metrics as a function of time, i.e., we plot the average slowdown perceived by the end user during each time interval corresponding to K requests. Since the system operates under transient overload conditions and is clearly not in steady state, our experiments focus on examining AdaptLoad's ability to respond to sudden bursts of arrivals and quickly serve as many requests as possible, as efficiently as possible.

What is the policy sensitivity to different hardware speeds? To address the common belief that replacing the servers with much faster ones solves the problem of transient overloads, we run our simulations assuming either "fast" or "slow servers". This is equivalent, in turn, to considering "low" and "high" load, respectively, and allows us

[9]We also experimented using the Join Shortest Queue (JSQ) policy but we do not report the results because they are consistently inferior to those for JSWQ.

to comment on AdaptLoad's ability to quickly flush backed-up queues.

Does the policy achieve equal utilization across servers? Since AdaptLoad bases its boundaries on knowledge of the distribution of requested file sizes, we examine the per-server utilization as a function of time and comment on the policy's ability to distribute load effectively.

Does AdaptLoad *treat short jobs differently from long jobs?* This question refers to the policy's *fairness*. To measure the responsiveness of the system, we report the average request slowdown of the classes of requests defined by the request sizes intervals.

Does AdaptLoad *scale well with respect to the cluster size?* Since AdaptLoad's ability to balance the load is a function of an effective mapping of different file sizes to specific servers, we explore the algorithm's scalability by running simulations using the same trace data but on an increased number of back-end servers.

Can we improve AdaptLoad*'s performance with smarter parameterization?* We address this issue by elaborating on alternative ways to use previous requests to predict the future ones. We introduce a new version of the algorithm based on a geometrically discounted history of the request sizes and report on its effectiveness.

7.6.4 AdaptLoad vs. JWSQ

For the first set of experiments we considered a cluster consisting of four back-end servers. Results are reported for JSWQ and AdaptLoad with $K = 50{,}000$. Figure

7.19 shows the average request slowdown in the cluster during the two-day trace under "low" and "high" load in the system. The average request slowdown closely follows the arrival intensities of Figure 7.17(a)-(b). During the "quiet" early morning to early afternoon hours, JSWQ does better than ADAPTLOAD, especially under low load. This is because the load in the system is so low that there is always an idle server for an incoming request. ADAPTLOAD, whose scheduling decisions are instead based exclusively on precomputed boundaries, may direct a request to a server that is already busy even when an idle server is available. During the periods of transient overload, however, we see a very different behavior: ADAPTLOAD greatly outperforms JSWQ and it consistently achieves lower average slowdowns (a log-scale is used for the vertical axis of the average request slowdown plots). During the most overloaded times, ADAPTLOAD is able to return faster to acceptable slowdown levels (see Figure 5(b)).

Figure 7.20 shows the utilization of each server across time for ADAPTLOAD and JSWQ, under high load and for the same two-days period as in Figure 7.19. JSWQ achieves a more uniform utilization across the four servers, while the per-server utilization of ADAPTLOAD has more variability, a direct effect of the algorithm's parameterization. Yet, even though the boundaries used are not optimal, ADAPTLOAD still achieves significantly better performance during overload periods.

Next, we consider the question of which requests are penalized most under ADAPTLOAD and JSWQ. Figure 7.21 illustrates the slowdown for various ranges of requests

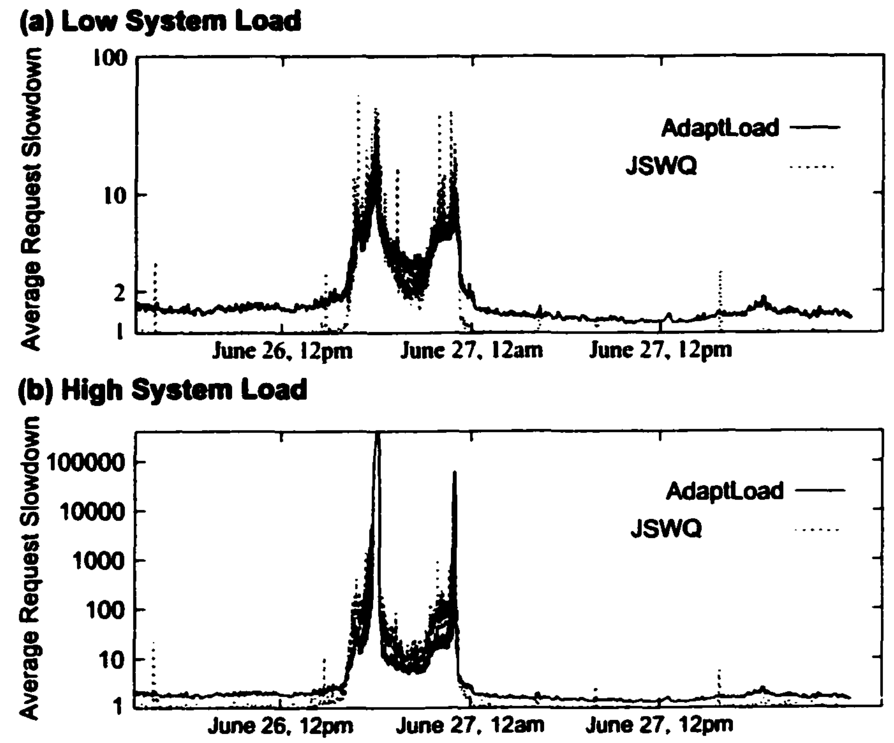

Figure 7.19: Average slowdown for ADAPTLOAD and JSWQ as a function of time, under either low or high load (four servers).

sizes for June 26, at 22:42:22, for the two policies, under either low or high load. The bar graph confirms that, with JSWQ, short requests (which account for the major portion of the requests) are penalized most because they tend to get blocked in the queue behind large ones. This effect is apparent under either load, but is more pronounced under high load, as apparent from Figure 7.21(b). Instead, ADAPTLOAD manages to consistently maintain small slowdowns for nearly all classes of requests, improving the overall system performance.

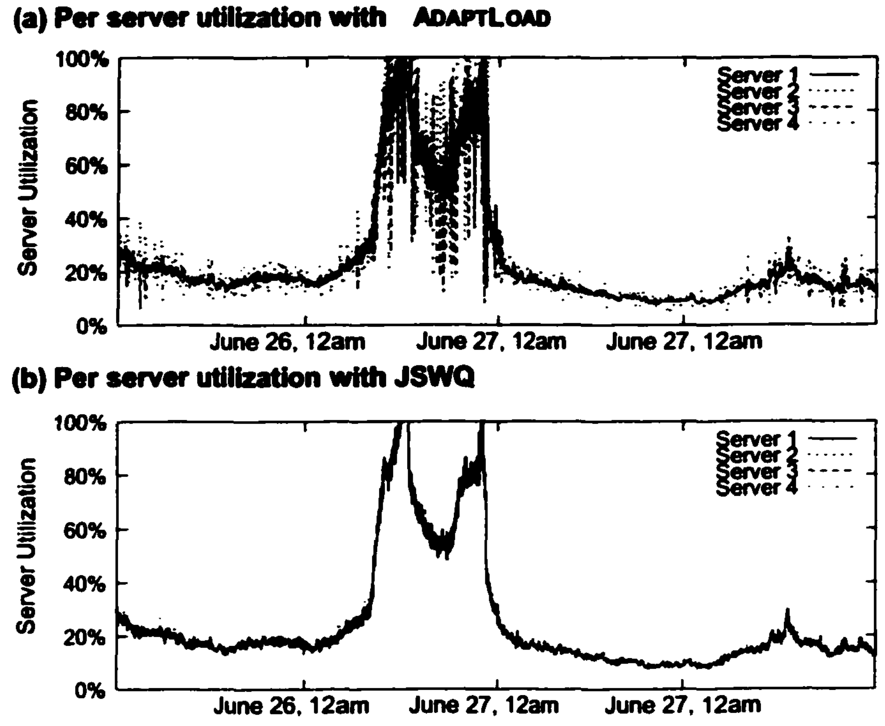

Figure 7.20: Server utilization with ADAPTLOAD or JSWQ as a function of time, under high load (four servers).

7.6.5 Scalability of ADAPTLOAD

To examine ADAPTLOAD's scalability as a function of the number of back-end servers, we double the number of servers from four to eight. We focus on the policy performance using the trace data of June 27 alone, a day showing more uniform arrival intensity but also more variable service demands when compared with June 26.

Figure 7.22 shows the request slowdown with the ADAPTLOAD and JSWQ policies as a function of time, under either low or high load. In either case, JSWQ cannot process the requests effectively, resulting in an expected slowdown curve with more "spikes" than ADAPTLOAD. The system reaches saturation around 6 pm with either

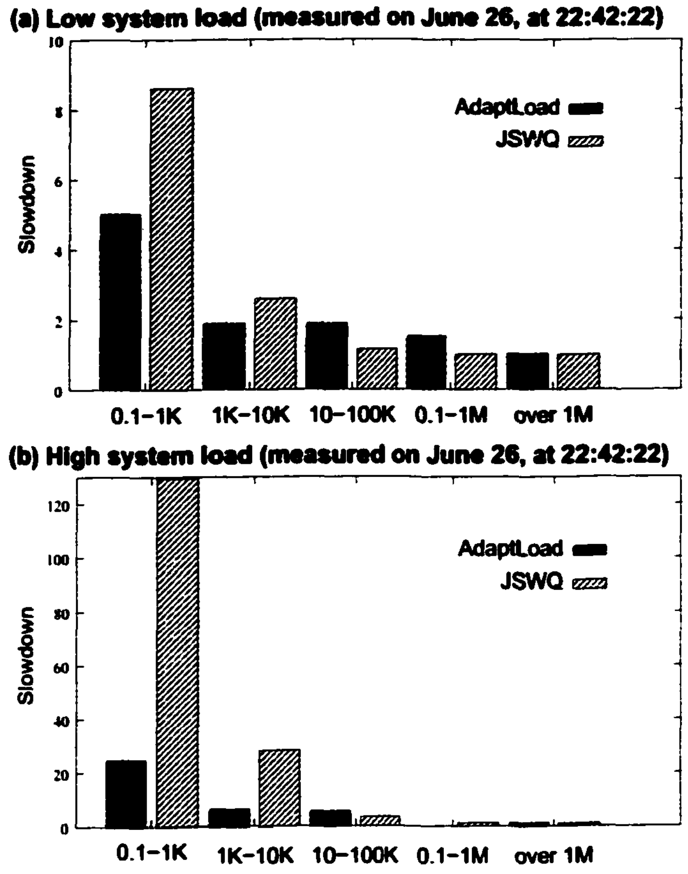

Figure 7.21: Slowdown by class of request sizes under either low or high load (four servers).

policy, but AdaptLoad recovers much faster than JSWQ. JSWQ is marginally better than AdaptLoad when the overall load is low, but substantially worse under overloads.

Figure 7.23(a) illustrates the request slowdown over the spectrum of request sizes with eight servers, for June 27 at 11:43:18, under low load. In this case, JSWQ performs slightly better than AdaptLoad, but note that the range of the y-axis is very narrow, from 0 to 2.5. This is a direct effect of the fact that the largest requests are quite rare, but our conservative estimation of boundaries allocates "too many"

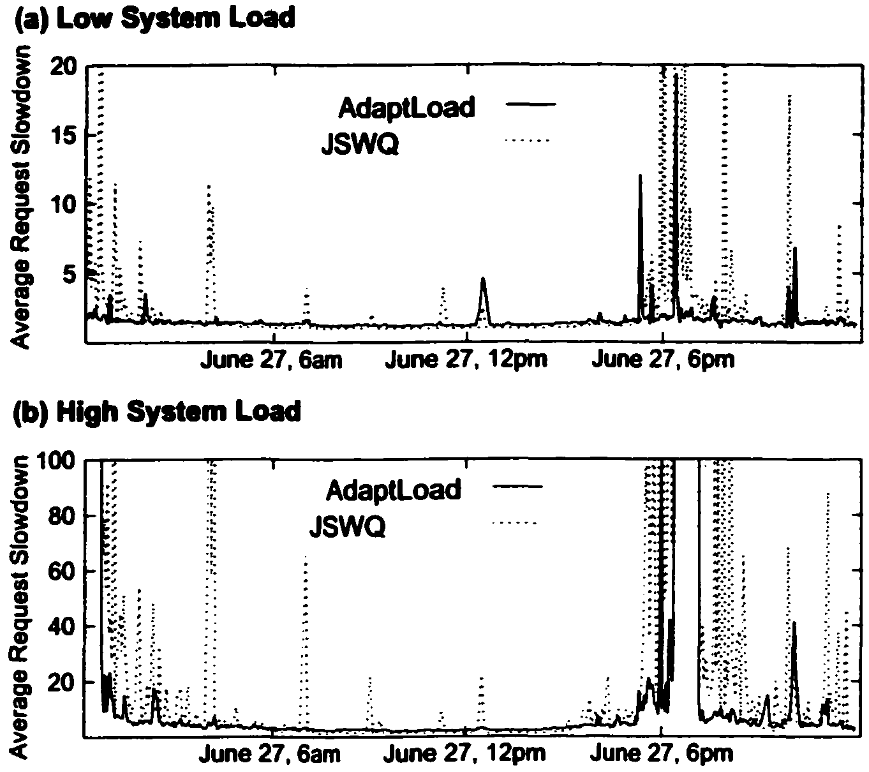

Figure 7.22: Average request slowdown of ADAPTLOAD and JSWQ policies under either low or high load (eight servers).

resources for the large requests, resulting in a reduction of the resources available to satisfy requests smaller than 1 MByte.

Figure 7.23(b) presents the per-class slowdown for June 27 at 20:37:32, under high load. Because of the presence of the whole range of requests in the K requests that correspond to the selected measurement point, ADAPTLOAD greatly outperforms JSWQ for all request ranges with the exception of the [0.1 – 1] MByte range. Even for cases where the number of servers per cluster is large but only few popular files in the discrete data histogram contribute to the majority of the distribution, these few files are served by multiple servers, and ADAPTLOAD still maintains high performance.

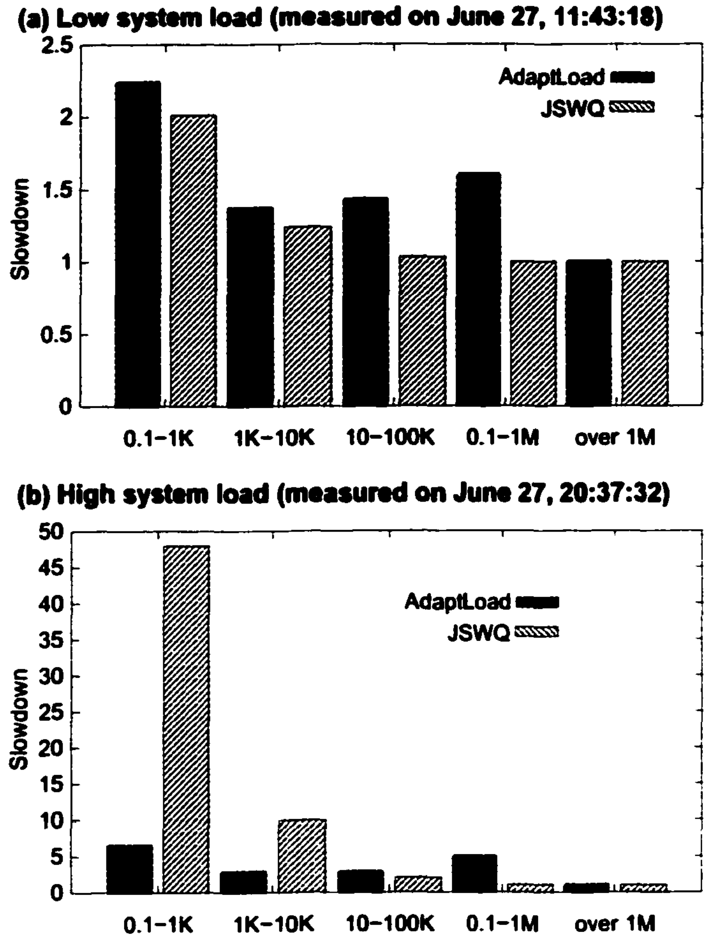

Figure 7.23: Slowdown by class of request sizes under either low or high load (eight servers).

7.6.6 Improving AdaptLoad

Recall that AdaptLoad's ability to balance the load is heavily influenced by the ability of predicting the distribution of the upcoming requests, based on knowledge about the past K requests. In this section, we propose an alternative approach that uses information about the entire past to increase policy responsiveness.

As with the basic AdaptLoad algorithm, we partition the flow of requests into batches of K requests and use a DDH to recompute the boundaries of every batch. However, instead of considering only the last batch, we now use information about *all* batches seen so far. Let O_i be a vector representing the DDH *observed* in the i^{th} batch. Then, the DDH U_i *used* to allocate a batch is obtained as a *geometrically discounted* weighted sum of all the previously observed batches:

$$U_{i+1} = \frac{\sum_{j=0}^{i} \alpha^{i-j} O_j}{\sum_{j=0}^{i} \alpha^{i-j}} = \frac{(1-\alpha)O_i + (\alpha-\alpha^{i+1})U_i}{1-\alpha^{i+1}}.$$

Where the positive coefficient α, $0 \le \alpha \le 1$, controls the rate at which memory of the past decreases in importance (the case $\alpha = 0$ corresponds to the algorithm previously presented, where only the last batch i is used to compute U_{i+1}; the case $\alpha = 1$ corresponds to giving the same weight to all batches). Since this version of ADAPTLOAD takes into consideration the whole history of the workload, K can be smaller comparing to the basic version of the policy, thus allowing for faster adaptations to workload changes. For any given trace and value of K, it is possible to find an *a posteriori* value of α providing nearly optimal performance: obviously, as a trend, the larger K, the smaller α. The α values determine how much of the past should be used to predict the future load.

ADAPTLOAD proves not to be very sensitive to (K, α) pairs. Table 7.2 reports α values that result in the lowest maximum observed queue length, for various K values and under either low or high load. For the computation of K we used the same two days of the trace as in the previous experiments. These values were found using an exhaustive search. Our exhaustive search indicated that it is possible to find a range of α's providing nearly-optimal performance. Experimenting with a fixed $K = 512$ and values within the range $[\alpha - 0.2, \alpha + 0.1]$ indicated that performance is very close to that with optimal α.

K	α for low load	α for high load
32768	0.2986328	0.0750000
8192	0.6998535	0.6228516
1024	0.9835938	0.9374023
512	0.9820313	0.9750000
256	0.9281250	0.9875000

Table 7.2: Optimal α values as a function of K, under high or low load.

Finally, we turn to the performance improvements with geometrically discounted history. Figures 7.24 illustrates the expected slowdown as a function of time for two versions of ADAPTLOAD: the basic one that considers only the immediate previous history, and one that uses all prior history using the optimal value of α. The selected time period are the peak hours of June 26, and we explored both low and high load. In both cases, the version of ADAPTLOAD with geometrically discounted history performs much better.

7.7 Chapter summary

In this chapter, we described how to analyze performance of Web servers using matrix-analytic methods. We fitted the service process of a Web server into a PH distribution and modeled the Web server as an M/PH/1 queue, which allowed us to use ETAQA

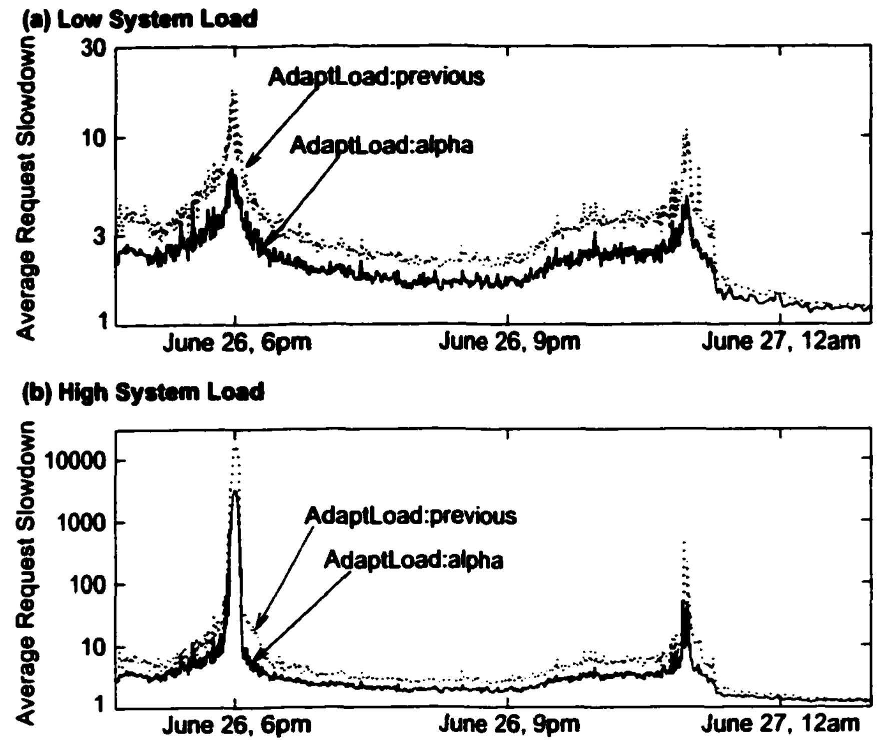

Figure 7.24: Average request slowdown for AdaptLoad under either low or high load (eight servers) with ($K = 512$, $\alpha = 0.92$).

for its performance analysis. Based on the results that we obtained from our analysis, we proposed EquiLoad, a content-aware load balancing policy for clustered Web servers, and AdaptLoad, its adaptive version. Our policies ensures that the expected load in each server, measured in "bytes to be transferred", is the same. Both EquiLoad and AdaptLoad achieve the desirable objective of maintaining high-degree of homogeneity in the requests allocated to each distinct server, hence operating as locality-aware policies. Our analysis showed that our policies sustain well the high load and the high variability in the cluster. AdaptLoad performs well even under swift fluctuations of the workload characteristics.

Chapter 8

Conclusions and future work

In this dissertation, we presented a set of modeling techniques for performance analysis of complex computer systems. In the following, we summarize the contributions of this dissertation.

- We proposed the D&C EM and D&C MM [79, 80, 23] parameterization techniques that approximate highly-variable data sets with PH distributions. First, we partition the data sets into subsets based on either their first moment (i.e., expected value) or their second moment (i.e., coefficient of variation), then we fit each subset into PH distributions using either the EM algorithm or the method of moments. The divide-and-conquer approach that we apply in both D&C EM and D&C MM increases their fitting accuracy and their computational efficiency. We evaluated the accuracy of D&C EM and D&C MM from the statistics and the queueing systems perspective.

- We captured long-range dependence in data sets using Hidden Markov models and PH distributions in a hierarchical fashion [86]. We evaluated the accuracy of this fitting method from the queueing system perspective.

- We prepared a survey on matrix-analytic techniques for solution of M/G/1-type, GI/M/1-type, and QBD processes [83]. We derived the matrix-analytic solution methods from first principles using stochastic complementation and illustrated the main concepts via simple examples.

- We developed ETAQA, a new aggregate matrix-analytic technique, that provides exact solutions for QBD, M/G/1-type, and GI/M/1-type e processes [21, 22, 81]. ETAQA computes an exact aggregate steady state probability distribution and a set of exact measures of interest. Detailed complexity analysis and experimental results demonstrate the computational efficiency and the numerical stability of the method.

- We developed and made available to the community a software tool, MAM-Solver[1], which provides implementations of classic and recent matrix-analytic methods, including the ETAQA methodology, for the solution of QBD, GI/M/1-type, and M/G/1-type processes [82].

- We developed a new technique for the exact solution of a restricted class of GI/G/1-type Markov processes [85]. Such processes exhibit both M/G/1-type and GI/M/1-type patterns and cannot be solved exactly with existing techniques. The proposed methodology uses decomposition to separate the M/G/1-

[1]Details available at `http://www.cs.wm.edu/MAMSolver/`

type and GI/M/1-type patterns, solves them independently, and aggregates the results to generate the final solution, i.e., the stationary probability vector.

- We demonstrated the applicability of our modeling techniques by evaluating the performance of load balancing policies in clustered Web servers. We proposed a size-based policy that assigns the incoming requests to the cluster based on their sizes [84].

- We proposed EQUILOAD, a load balancing policy in clustered Web servers [23]. EQUILOAD assigns each server of the cluster to serve a different pre-determined range of request sizes. Numerous experiments demonstrate that EQUILOAD outperforms traditional load balancing policies and improves user perceived performance.

- We improved EQUILOAD to adapt to the transient load conditions commonly experienced by clustered Web servers and proposed ADAPTLOAD [87]. ADAPTLOAD maintains good performance under conditions of transient overloads.

8.1 Future directions

Future plans consists of extensions of the work presented in this dissertation, as well as exploration of new problems raised while developing our new modeling techniques and load balancing policies. In the following, we outline future research plans:

- D&C EM and D&C MM, proposed in Chapter 4, fit data sets into PH distributions in a divide and conquer fashion. The major differences between these two techniques are
  - the fitting algorithms used, i.e., D&C EM uses the EM algorithm and D&C MM uses the moment matching techniques
  - the criteria for partitioning the data set, i.e., D&C EM partitions the data in equally variable subsets and D&C MM partitions the data in subsets with equal expected value.

  We plan to work on a rigorous comparison between these two techniques and identify which are the benefits of using the available partitioning criteria and specific fitting algorithms. Based on such analysis, we intend to propose fitting techniques that are fast and accurate on capturing complex data characteristics.

- In addition to providing fitting techniques for data sets with complete and non-complete monotone CDHs, we will investigate the possibility of fitting multi-modal workloads into PH distributions. Evidence shows that such multi-modal workloads exist in communication systems.

- The ETAQA methodology, proposed in Chapter 5, presents an aggregation-based matrix-analytic technique for the solution of Markov processes with repetitive structure. We demonstrated via initial experimenting that ETAQA is numerically stable. We plan to further work in this direction and design numerical experiments that aim to the systematic evaluation of the numerical stability of matrix-analytic methods.

- Currently ETAQA provides solution for infinite Markov chains with repetitive structure. We plan to extend ETAQA for the solution of M/G/1/K-type processes, i.e., finite Markov chains with repetitive structure. With this extension, we aim to use ETAQA for the analysis of queueing systems with finite buffers, commonly encountered in computer systems.

- We will investigate the possibility of using the same aggregation technique as in ETAQA for the exact and/or approximate solution of queueing models with multiple job classes and/or multiple servers in the service center.

- We will modify ADAPTLOAD to also consider dynamic requests in the clustered Web server. We will evaluate different ways to assign requests of unknown sizes to servers of the cluster while maintaining high system performance.

- We plan to extend ADAPTLOAD to accommodate requests for different classes of service. Our objective is to evaluate if it is more beneficial to partition the servers of the cluster in dedicated subclusters for specific classes of requests, or differentiate service within a single server by allowing high priority requests to use more resources within the server than low priority requests.

- Currently the ADAPTLOAD algorithm is centralized, i.e., we collect all the information about the requests in the cluster in one single device (i.e., either dispatcher or a Web server). This device determines the ADAPTLOAD's parameters and distributes them to the servers of the cluster. We will propose a variation of ADAPTLOAD that allows distributed computation of its parameters, i.e., each server will exchange information only with its pre-determined neighbors and adapt its own size boundaries based on the workload seen by the server itself and its neighbors.

Appendix A

Feldman-Whitt Fitting Algorithm

The Feldmann-Whitt algorithm [30] attempts to fit various regions of the distribution with exponential phases in a recursive manner. At each step, the fitted exponential component is subtracted from the distribution, such that each component focuses on a specific portion of the random variable values, increasingly closer to 0. If there are enough exponential components, the algorithm manages to closely approximate a heavy-tail distribution in the area of primary interest. The algorithm takes a heavy-tail distribution with cumulative distribution (cdf) $F(x)$ and complementary cumulative distribution (ccdf) $F^c(x)$ over a sufficiently large range $[c_k, c_1]$ and fits it to a hyperexponential distribution H_k. The steps of the fitting algorithm are:

step 1: Choose a number k of exponential components and k arguments so as to match the distribution quantiles, $0 < c_k < c_{k-1} < ... < c_1$. The ratios c_i/c_{i+1} are assumed to be sufficiently large, such that $1 < b < c_1/c_{i+1}$ for all i.

step 2: μ_1 and p_1 should match the ccdf $F^c(x)$ at the arguments c_1 and bc_1, therefore

$$\mu_1 = \frac{1}{(b-1)c_1} \ln(F^c(c_1)/F^c(bc_1)),$$

$$p_1 = F^c(c_1)e^{-\mu_1 c_1}.$$

In this procedure it is assumed that μ_i is sufficiently larger than μ_1 for all $i \geq 2$, that the final approximation satisfies

$$\sum_{i=1}^{k} \approx p_1 e^{-\mu_1 t}, t \geq c_1.$$

step 3: For $2 \leq i \leq k$, the following are defined:

$$F_i^c(c_i) = F_{i-1}^c(c_i) - \sum_{j=1}^{i-1} p_j e^{-\mu_j c_i},$$

$$F_i^c(bc_i) = F_{i-1}^c(bc_i) - \sum_{j=1}^{i-1} p_j e^{-\mu_j bc_i},$$

where $F_1^c(x) = F^c(x)$. As in step 2, μ_i and p_i for $2 \leq i \leq k-1$ are defined as

$$\mu_i = \frac{1}{(b-1)c_i} \ln(F_i^c(c_i)/F_i^c(bc_i)),$$

$$p_i = F_i^c(c_i)e^{-\mu_1 c_i}.$$

step 4: The last parameter pair (μ_k, p_k) is computed using:

$$p_k = 1 - \sum_{j=1}^{k-1} p_j,$$

$$\mu_k = \frac{1}{c_k}\ln(p_k/F_k^c(c_k)).$$

The Feldmann-Whitt algorithm is suitable for distributions, whose PDF is complete monotone decreasing, but, generally, it works for any other distribution. This algorithm is proposed for fitting one continuous distribution into another.

Appendix B

(B)MAP/PH/1 queues

A MAP/PH/1 represents a single server queue that has MAP arrival process and PH service process. The MAP/PH/1 queue is a quasi birth-death process whose matrices can be computed using the matrix parameters that define the MAP and the PH processes of the MAP/PH/1 queue. Let the MAP describtors to be $(\mathbf{D}_0, \mathbf{D}_1)$ of order m_A and the PH service time parameters to be $(\boldsymbol{\tau}, \mathbf{T})$ of order m_B. The infinitesimal generator matrix $\mathbf{Q}_{MAP/PH/1}$ for the corresponding MAP/PH/1 queue has a structure given by

$$\mathbf{Q}_{MAP/PH/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad (B.1)$$

where each of its matrices is defined as follows

$$\begin{array}{ll} \widehat{\mathbf{L}} = \mathbf{D}_0, & \mathbf{L} = \mathbf{I}_{m_A} \otimes \mathbf{T} + \mathbf{D}_0 \otimes \mathbf{I}_{m_B}, \\ \widehat{\mathbf{B}} = \mathbf{I}_{m_A} \otimes \mathbf{T}^0, & \mathbf{B} = \mathbf{I}_{m_A} \otimes \mathbf{T}^0 \boldsymbol{\tau}, \\ \widehat{\mathbf{F}} = \mathbf{D}_1 \otimes \boldsymbol{\tau}, & \mathbf{F} = \mathbf{D}_1 \otimes \mathbf{I}_{m_B}. \end{array} \qquad (B.2)$$

where $\mathbf{I}_k$ is the order k identity matrix, $\mathbf{T}^0 = -\mathbf{T} \cdot \mathbf{e}$, and $\otimes$ denotes the Kroneker product.

A BMAP/PH/1 represent a single server queue that has BMAP arrival process and PH service process. The BMAP/PH/1 queue results in a M/G/1-type process. Let the BMAP parameteres be $\mathbf{D}_i$ for $i \geq 0$ and PH parameters be $(\boldsymbol{\tau}, \mathbf{T})$ of order m_A and m_B, respectively. The infinitesimal generator matrix $\mathbf{Q}_{BMAP/PH/1}$ for the corresponding BMAP/PH/1 queue is given by

$$\mathbf{Q}_{BMAP/PH/1} = \begin{bmatrix} \widehat{\mathbf{L}} & \widehat{\mathbf{F}}^{(1)} & \widehat{\mathbf{F}}^{(2)} & \widehat{\mathbf{F}}^{(3)} & \widehat{\mathbf{F}}^{(4)} & \cdots \\ \widehat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad \text{(B.3)}$$

where each of its matrices is defined as follows

$$\begin{array}{ll} \widehat{\mathbf{L}} = \mathbf{D}_0, & \mathbf{L} = \mathbf{I}_{m_A} \otimes \mathbf{T} + \mathbf{D}_0 \otimes \mathbf{I}_{m_B}, \\ \widehat{\mathbf{B}} = \mathbf{I}_{m_A} \otimes \mathbf{T}^0, & \mathbf{B} = \mathbf{I}_{m_A} \otimes \mathbf{T}^0 \boldsymbol{\tau}, \\ \widehat{\mathbf{F}}^{(i)} = \mathbf{D}_i \otimes \boldsymbol{\tau}, & \mathbf{F}^{(i)} = \mathbf{D}_i \otimes \mathbf{I}_{m_B}, \quad i \geq 1, \end{array} \qquad \text{(B.4)}$$

where $\mathbf{I}_k$ is the order k identity matrix, $\mathbf{T}^0 = -\mathbf{T} \cdot \mathbf{e}$, and $\otimes$ denotes the Kroneker product.

Appendix C

Newton-Raphson Fitting Technique

This methodology can be used to describe one distribution with exponential phases by matching the moments of the distribution functions [98]. The Newton-Raphson technique is based on an iterative algorithm. The initial guessed solution is used to generate the series of intermediate solutions which eventually converge to the exact one.

Fitting data into a distribution means that the PDF, CDD, and the moments of the distribution should closely approximate the empirical distribution, the cumulative empirical distribution, and any of the moments of the data set, respectively. Analoguously a distribution can be fitted into another distribution. The resulting distribution, commonly called the estimated distribution is determined once its parameters are known. If we fit a data set into an hyperexponential distribution, H_k, we need to estimate $2k - 1$ parameters. The technique of matching the moments solves

$2k-1$ different equations in order to compute these parameters. It is assumed that from the data, or the original distribution we can compute at least the first $2k-1$ moments, which we denote by M_i $i = 1, ..., 2k-1 = r$.

If the goal is to match r moments then the following r functions are computed in each step of the algorithm:

$$\begin{array}{rcccl} \phi_1 & = & \phi_1(\mathbf{X}) & = & m_1(\mathbf{X}) - M_1 \\ \phi_2 & = & \phi_2(\mathbf{X}) & = & m_2(\mathbf{X}) - M_2 \\ & & \cdots & & \\ \phi_r & = & \phi_r(\mathbf{X}) & = & m_r(\mathbf{X}) - M_r \end{array}. \tag{C.1}$$

where $\mathbf{X}$ is the column vector $(x_1.x_2, ...x_r)^T$ of the r parameters that need to be computed and $m_i(\mathbf{X})$ is the i^{th} moment of the phase-type distribution. The conditions of exact match of the first r moments are

$$\phi_1 = \phi_2 = ... = \phi_r = 0. \tag{C.2}$$

Let $\mathbf{X}^i = (x_1^i, x_2^i, ...x_r^i)^T$ be the vector of parameters on the i^{th} stage of approximation, and $\mathbf{\Phi}$ the column vector $\mathbf{\Phi} = (\phi_1^i, \phi_2^i, ...\phi_r^i)^T$. The correction vector for parameter $\Delta\mathbf{X}^i = (\Delta x_1^i, \Delta x_2^i, ...\Delta x_r^i)^T$ is computed from the matrix equation

$$\mathbf{\Phi}(\mathbf{X}^i) + \mathbf{\Phi}'(\mathbf{X}^i)\Delta\mathbf{X}^i = \mathbf{0}, \tag{C.3}$$

where $\mathbf{\Phi}'(\mathbf{X}^i)$ is the Jacobian matrix of $\mathbf{\Phi}$ at $\mathbf{X}^i$

$$\mathbf{\Phi}'(\mathbf{X}^i) = \begin{bmatrix} \frac{\partial\phi_1}{\partial x_1}(\mathbf{X}^i) & \frac{\partial\phi_1}{\partial x_2}(\mathbf{X}^i) & \cdots & \frac{\partial\phi_1}{\partial x_r}(\mathbf{X}^i) \\ \frac{\partial\phi_2}{\partial x_1}(\mathbf{X}^i) & \frac{\partial\phi_2}{\partial x_2}(\mathbf{X}^i) & \cdots & \frac{\partial\phi_2}{\partial x_r}(\mathbf{X}^i) \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial\phi_r}{\partial x_1}(\mathbf{X}^i) & \frac{\partial\phi_r}{\partial x_2}(\mathbf{X}^i) & \cdots & \frac{\partial\phi_r}{\partial x_r}(\mathbf{X}^i) \end{bmatrix}. \tag{C.4}$$

Eq.(C.3) is solved using Gaussian elimination [107]. The improved parameter values are obtained by $\mathbf{X} = \mathbf{X}^i + \Delta\mathbf{X}^i$. $\mathbf{\Phi}(\mathbf{X}^i)$ is calculated directly from the first r moments of the phase-type distribution when $\mathbf{X} = \mathbf{X}^i$. The above steps are repeated until the a solution is obtained within a certain accuracy.

Appendix D

Examples of MAMSolver input

In this Appendix we present MAMSolver input examples for simple Markov chains of QBD, GI/M/1, and M/G/1 type, to illustrate how to use our matrix-analytic methods tool. In the following we characterize each of the queueing systems, whose MAMSolver inputs we present here.

1. M/M/1 queue, embeded in a CTMC, with arrival rate 2.0 and service rate 3.0 (Table D.1).

2. M/Cox$_2$/1 queue, embeded in a CTMC, with arrival rate 2.0 and 2-phase Coxian service process (Table D.2). The parameters $(\boldsymbol{\tau}, \mathbf{T})$ of the Coxian distribution, defined in Table 2.1, in this particular example are:

$$\boldsymbol{\tau} = [1,\ 0] \quad \mathbf{T} = \begin{bmatrix} -10.0 & 4.0 \\ 0.0 & 3.0 \end{bmatrix}.$$

3. M/BMAP$_1$/1 queue, embeded in a CTMC of GI/M/1-type, with arrival rate 2.0 and BMAP$_1$ service process (Table D.3). The BMAP$_1$ service process in

this example has batch size 3, which means that the BMAP$_1$ is defined by ($\mathbf{D}_0$, $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$) as follows

$$\mathbf{D}_0 = [-6.0], \quad \mathbf{D}_1 = [3.0], \quad \mathbf{D}_2 = [2.0], \quad \mathbf{D}_3 = [1.0].$$

4. BMAP$_1$/Hr$_3$/1 queue, embeded in a CTMC of M/G/1-type, with Batch Markovian Arrival process and 3-phase hyperexponential service (Table D.4). The BMAP$_1$ arrival process in this example has batch size 4, which means that the BMAP$_1$ is defined by ($\mathbf{D}_0$, $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, $\mathbf{D}_4$) as follows

$$\mathbf{D}_0 = [-3.75], \quad \mathbf{D}_1 = [2.0], \quad \mathbf{D}_2 = [1.0], \quad \mathbf{D}_3 = [0.50], \quad \mathbf{D}_3 = [0.25].$$

The 3-phase hyperexponential is defined by $(\boldsymbol{\tau}, \mathbf{T})$ as follows

$$\boldsymbol{\tau} = [0.75, \ 0.2, \ 0.05] \quad \mathbf{T} = \begin{bmatrix} -10.0 & 0.0 & 0.0 \\ 0.0 & -12.0 & 0.0 \\ 0.0 & 0.0 & -16.0 \end{bmatrix}.$$

The matrices $\widehat{\mathbf{L}}$, $\widehat{\mathbf{B}}^{(i)}$, $\widehat{\mathbf{F}}^{(i)}$, $\mathbf{L}$, $\mathbf{B}^{(i)}$, $\mathbf{F}^{(i)}$ for $1 \le i \le l$ are determined using the rules outlined in Appendix B. We note that the explanations given on the right column of each input example table are not part of the actual input file.

1		size of boundary portion m
1		size of repetitive portion n
1		batch length l
0.000000000001		numerical accuracy of solution
-2.0		$\hat{\mathbf{L}}$
3.0		$\hat{\mathbf{B}}$
2.0		$\hat{\mathbf{F}}$
3.0		$\mathbf{B}$
-5.0		$\mathbf{L}$
2.0		$\mathbf{F}$

Table D.1: Input for an M/M/1 queue

1		size of boundary portion m
2		size of repetitive portion n
1		batch length l
0.000000000001		numerical accuracy of solution
-2.0		$\hat{\mathbf{L}}$
6.0		$\hat{\mathbf{B}}$
3.0		
2.0	0.0	$\hat{\mathbf{F}}$
6.0	0.0	$\mathbf{B}$
3.0	0.0	
-12.0	4.0	$\mathbf{L}$
0.0	-5.0	
2.0	-0.0	$\mathbf{F}$
0.0	-2.0	

Table D.2: Input for an M/Cox$_2$/1 queue

1	size of boundary portion m
1	size of repetitive portion n
3	batch length l
0.000000000001	numerical accuracy of solution
-2.0	$\widehat{\mathbf{L}}$
2.0	$\widehat{\mathbf{F}}$
6.0	$\widehat{\mathbf{B}}^{(1)}$
3.0	$\widehat{\mathbf{B}}^{(2)}$
1.0	$\widehat{\mathbf{B}}^{(3)}$
-8.0	$\mathbf{L}^{(1)}$
3.0	$\mathbf{B}^{(1)}$
2.0	$\mathbf{B}^{(2)}$
1.0	$\mathbf{B}^{(3)}$
2.0	$\mathbf{F}$
-8.0	$\mathbf{L}$
3.0	$\mathbf{B}^{(1)}$
2.0	$\mathbf{B}^{(2)}$
1.0	$\mathbf{B}^{(3)}$

Table D.3: Input for an DTMC of GI/M/1-type

1			size of boundary portion m
3			size of repetitive portion n
4			batch length l
0.000000000001			numerical accuracy of solution
-3.75			$\hat{\mathbf{L}}$
10.0			$\hat{\mathbf{B}}$
12.0			
16.0			
2.0	0.0	0.0	$\hat{\mathbf{F}}^{(1)}$
1.0	0.0	0.0	$\hat{\mathbf{F}}^{(2)}$
0.5	0.0	0.0	$\hat{\mathbf{F}}^{(3)}$
0.25	0.0	0.0	$\hat{\mathbf{F}}^{(4)}$
7.5	2.0	0.5	$\mathbf{B}$
9.0	2.4	0.6	
12.0	3.2	0.8	
-13.75	0.0	0.0	$\mathbf{L}$
0.0	-15.75	0.0	
0.0	0.0	-19.75	
2.0	0.0	0.0	$\mathbf{F}^{(1)}$
0.0	2.0	0.0	
0.0	0.0	2.0	
1.0	0.0	0.0	$\mathbf{F}^{(2)}$
0.0	1.0	0.0	
0.0	0.0	1.0	
0.5	0.0	0.0	$\mathbf{F}^{(3)}$
0.0	0.5	0.0	
0.0	0.0	0.5	
0.25	0.0	0.0	$\mathbf{F}^{(4)}$
0.0	0.25	0.0	
0.0	0.0	0.25	

Table D.4: Input for an $\mathrm{BMAP}_1/\mathrm{Hr}_3/1$ queue

Bibliography

[1] A. T. ANDERSEN. *Modeling of Packet Traffic With Matrix Analytic Methods.* IMM Dept., Technical University Denmark, 1995. PhD thesis.

[2] D. ANDERSEN, T. YANG, AND O. IBARRA. Toward a scalable distributed WWW server on workstation clusters. *Journal of Parallel and Distributed Computing,* 42:91–100. September 1997.

[3] M. ARLITT AND T. JIN. Workload characterization of the 1998 World Cup Web site. Technical report, Hewlett-Packard Laboratories, September 1999.

[4] M. ARLITT AND C.L. WILLIAMSON. Web server workload characterization. the search for invariants. In *Proceedings of ACM SIGMETRICS Conference,* pages 126–138, Philadelphia, PA, May 1996.

[5] M. ARON, D. SANDERS, P. DRUSCHEL, AND W. ZWAENEPOEL. Scalable content-aware request distribution in cluster-based network servers. In *Proceedings of Annual USENIX Technical Conference,* San Diego, CA, June 2000.

[6] S. ASMUSSEN, O. NERMAN, AND M. OLSON. Fitting Phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics,* 23:419–441, 1996.

[7] P. BARFORD, A. BESTAVROS, A. BRADLEY, AND M. E. CROVELLA. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web, Special Issue on Characterization and Performance Evaluation,* 2(0):15–28, 1999.

[8] P. BARFORD AND M. E. CROVELLA. Generating representative web workloads for network and server performance evaluation. In *Proceedings of Performance / ACM SIGMETRICS Conference,* pages 151–160, Madison, WI, May 1998.

[9] N. G. BEAN, J.-M. LI, AND P. G. TAYLOR. Caudal characteristics of QBDs with decomposable phase spaces. In *Advances in Algorithmic Methods for Stochastic Models,* G. Latouche and P. Taylor, editors, pages 37–55. Notable Publications, 2000.

[10] J. BERAN. *Statistics for Long-Memory Processes*. Chapman & Hall. New York, 1994.

[11] D. A. BINI AND B. MEINI. Using displacement structure for solving non-skip-free M/G/1 type Markov chains. In *Advances in Matrix Analytic Methods for Stochastic Models*, A. S. Alfa and S. R. Chakravarthy, editors, pages 17–37, NJ, 1998. Notable Publications Inc.

[12] D. A. BINI, B. MEINI, AND V. RAMASWAMI. Analyzing M/G/1 paradigms through QBDs: the role of the block structure in computing the matrix G. In *Advances in Matrix Analytic Methods for Stochastic Models*, G. Latouche and P. Taylor, editors, pages 73–86, NJ, 2000. Notable Publications Inc.

[13] A. BOBBIO AND A. CUMANI. ML estimation of the parameters of a PH distribution in triangular canonical form. In *Computer Performance Evaluation*, G. Balbo and G. Serazzi, editors, pages 33–46. Elsevier Science Publishers, 1992.

[14] S. C. BORST, O. J. BOXMA, AND R. NUNEZ-QUEIJA. Heavy tails: The effect of the service discipline. In *Proceedings of TOOLS 2002: LNCS 2324*, T. Field, P. Harrison, J. Bradley, and U. Harder, editors, pages 1–30. Springer-Verlag, 2002.

[15] L. BREUER. Parameter estimation for a class of BMAPs. In *Advances in Algorithmic Methods for Stochastic Models*, G. Latouche and P. Taylor, editors, pages 87–97. Notable Publications, 2000.

[16] L. BREUER. An EM algorithm for Batch Markovian Arrival Processes and its comparison to a simpler estimation procedure. Technical Report 01-14, Department of Mathematics and Computer Science, University of Trier, Germany, 2001.

[17] V. CARDELLINI, M. COLAJANNI, AND P.S. YU. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 2(3):28–39, June 1999.

[18] V. CARDELLINI, M. COLAJANNI, AND P.S. YU. Redirection algorithms for load sharing in distributed web-server systems. In *Proceedings of the 19th IEEE International Conference on Distributed Computer Systems (ICDCS'99)*, pages 528–535, Austin, TX, June 1999.

[19] L. CHERKASOVA. Flex: Design and management strategy for scalable web hosting service. Technical Report HPL-1999-64R1, Hewlett-Packard Laboratories, May 1999.

[20] S.-H. CHIANG, R. K. MANSHARAMANI, AND M. K. VERNON. Use of application characteristics and limited preemption for run-to-completion parallel

processor scheduling policies. In *Proceedings of ACM SIGMETRICS Conference*, pages 33–44, Nashville, TN, May 1994.

[21] G. CIARDO, A.RISKA, AND E. SMIRNI. An aggregation-based solution method for M/G/1-type processes. In *Numerical Solution of Markov Chains*, B. Plateau, W. J. Stewart, and M. Silva, editors, pages 21–40, Zaragoza, Spain, September 1999. Prensas Universitarias de Zaragoza.

[22] G. CIARDO, W. MAO, A. RISKA, AND E. SMIRNI. ETAQA-MG1: An efficient technique for the analysis of M/G/1-type processes by aggregation. *Performance Evaluation Journal*, (second round of revision).

[23] G. CIARDO, A. RISKA, AND E. SMIRNI. EQUILOAD: a load balancing policy for clustered web servers. *Performance Evaluation*, 46:101–124, 2001.

[24] G. CIARDO AND E. SMIRNI. ETAQA: An efficient technique for the analysis of QBD-processes by aggregation. *Performance Evaluation*, 36-37:71–93, 1999.

[25] CISCO SYSTEMS INC. *Local Director*. http://www.cisco.com.

[26] A. E. CONWAY AND N. D. GEORGANAS. *Queueing Networks-Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation*. Computer Systems Series, MIT Press, 1989.

[27] P. J. COURTOIS. *Decomposability: Queueing and Computer System Applications*. ACM monograph series: Academic Press, New York, 1977.

[28] M.E. CROVELLA AND M.S. TAQQU. Estimating the heavy tail index from scaling properties. *Methodology and Computing in Applied Probability*, 1(1):55–79, 1999.

[29] J. N. DAIGE AND D. M. LUCANTONI. Queueing systems having phase-dependent arrival and service rates. In *Numerical Solution of Markov Chains*, J. W. Stewart, editor, pages 179–215, NY, 1991. Marcel Dekker.

[30] A. FELDMANN AND W. WHITT. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31(8):963–976, August 1998.

[31] J. D. FERGUSON. Variable duration models for speach. In *Proceedings of Symposium on Applications of Hidden Markov Models in Text and Speach*, pages 143–179, October 1980.

[32] H. R. GAIL, S. L. HANTLER, AND B. A. TAYLOR. Use of characteristic roots for solving infinite state Markov chains. In *Computational Probability*, W. K. Grassman, editor, pages 205–255, Boston, MA, 2000. Kluwer Academic Publishers.

[33] L. Golubchik and J. C. Lui. A fast and accurate iterative solution of a multiclass threshold-based queueing system with hysteresis. In *Proceedings of ACM SIGMETRICS Conference*, pages 196–206, Santa Clara, CA, June 2000.

[34] W. K. Grassman and D. A. Stanford. Matrix analytic methods. In *Computational Probability*, W. K. Grassman, editor, pages 153–204, Boston, MA, 2000. Kluwer Academic Publishers.

[35] R. Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *IEEE Journal on Selected Areas in Communications*, 19(2):203–211, 1991.

[36] M. Harchol-Balter, M.E. Crovella, and C.D. Murta. On choosing a task assignment policy for a distributed server system. In *Proceedings of Performance Tools '98, Lecture Notes in Computer Science, Volume 1469*, pages 231-242, Boston, MA, 1998. Springer Verlag.

[37] M. Harchol-Balter, M.E. Crovella, and S. S. Park. The case for SRPT scheduling in web servers. Technical Report MIT-LCS-TR-767, MIT Laboratory for Computer Science, October 1998.

[38] D. Heyman and D. Lucantoni. Modeling multiple IP traffic streams with rate limits. In *Proceedings of the 17th International Teletraffic Congress*, Brazil, December 2001.

[39] A. Horvath, G. Rozsa, and M. Telek. A MAP fitting method to approximate real traffic behaviour. In *Proceedings of the 8th IFIP Workshop on Performance Modelling and Evaluation of ATM & IP Networks*, G. Latouche and P. Taylor, editors, pages 32/1-12, Ilkley, UK, 2000.

[40] A. Horvath and M. Telek. Approximating heavy tailed behavior with phase type distribution. In *Advances in Algorithmic Methods for Stochastic Models*, G. Latouche and P. Taylor, editors, pages 191–214. Notable Publications, 2000.

[41] IBM Corporation. *IBM Interactive Network Dispatcher*. http://www.ics.raleigh.ibm.com/ics/isslearn.htm.

[42] M. A. Johnson and M. R. Taffe. Matching moments to phase distributions: Mixtures of Erlang distribution of common order. *Stochastic Models*, 5:711–743, 1989.

[43] L. Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley, 1975.

[44] A. Klemm, C. Lindemann, and M. Lohmann. Traffic modeling using the Batch Markovian Arrival process. In *Proceedings of TOOLS 2002: LNCS 2324*, T. Field, P. Harrison, J. Bradley, and U. Harder, editors, pages 92–110. Springer-Verlag, 2002.

[45] G. LATOUCHE. A simple proof for the matrix-geometric theorem. *Applied Stochastic Models and Data Analysis*, 8:25–29, 1992.

[46] G. LATOUCHE. Algorithms for infinite Markov chains with repeating columns. In *Linear Algebra, Markov Chains, and Queueing Models*, C. Meyer and R. J. Plemmons, editors, volume 48, pages 231-265. IMA Volumes in Mathematics and its Applications, Springer Verlag, 1993.

[47] G. LATOUCHE AND V. RAMASWAMI. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* SIAM, Philadelphia PA, 1999. ASA-SIAM Series on Statistics and Applied Probability.

[48] A. M. LAW AND W. D. KELTON. *Simulation Modeling and Analysis, Third Edition.* McGraw-Hill Inc., 2000.

[49] H. LEEMANS. *The Two-Class Two-Server Queueing Model with Nonpreemptive Heterogeneous Priority Structures.* K.U.Leuven, Belgium, 1998. PhD thesis.

[50] W. E. LELAND, M. S. TAQQU, W. WILLINGER, AND D. V. WILSON. On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2:1-15, 1994.

[51] J. D. C. LITTLE. A proof of the queuing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.

[52] Z. LIU, N. NICLAUSSE, AND C. JALPA-VILLANUEVA. Web traffic modeling and performance comparison between HTTP 1.0 and HTTP 1.1. In *Systems Performance Evaluation: Methodologies and Applications*, E. Gelenbe, editor, pages 177-189. CRC Press, 2000.

[53] D. M. LUCANTONI. *An algorithmic analysis of a communication model with retransmission of flawed messages.* Pitman, Boston, 1983.

[54] D. M. LUCANTONI. The BMAP/G/1 queue: A tutorial. In *Models and Techniques for Performance Evaluation of Computer and Communication Systems*, L. Donatiello and R. Nelson, editors, pages 330–358. Springer-Verlag, 1993.

[55] C. MCCANN, R. VASWANI, AND J. ZAHORJAN. A dynamic processor allocation policy for multiprogrammed shared memory multiprocessors. *ACM Transactions on Computer Systems*, 11(2):146–178, 1993.

[56] K. S. MEIER-HELLSTERN. A fitting algorithm for Markov-modulated Poisson processes having two arrival rates. *European Journal of Operations Research*, 29:370–377, 1987.

[57] B. MEINI. *Implementation of FFT-based version of Ramaswami's formula.* University of Pisa, Italy, 1997.

[58] B. MEINI. An improved FFT-based version of Ramaswami's formula. *Comm. Statist. Stochastic Models*, 13:223–238, 1997.

[59] B. MEINI. *Fast algorithms for the numerical solution of structured Markov chains.* Department of Mathematics, University of Pisa, Italy, 1998. PhD thesis.

[60] B. MEINI. Solving M/G/1-type Markov chains: Recent advances and applications. *Comm. Statist. Stochastic Models*, 14(1 - 2):479–496, 1998.

[61] C. D. MEYER. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31(2):240–271, 1989.

[62] C. MITCHELL AND L. JAMIESON. Modeling duration in a hidden Markov model with the exponential family. In *Proceedings of ICASSP '93*, pages 331–334, 1993.

[63] J. R. MOELLER. *Matrix-analytic Methods and on Collective Risk in Life Insurance.* Center for Mathematical Sciences and Mathematical Statistics, Lund University, Sweeden, 2000. PhD Thesis.

[64] R. NELSON. Matrix geometric solutions in Markov models: a mathematical tutorial. Technical Report RC 16777 (#742931), IBM T.J. Watson Research Center, Yorktown Heights, NY, April 1991.

[65] R. NELSON. *Probability, Stochastic Processes, and Queueing Theory.* Springer-Verlag, 1995.

[66] R. NELSON AND B.R. IYER. Analysis of a replicated data base. *Performance Evaluation*, 5:133–148, 1985.

[67] M. F. NEUTS. *Matrix-geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, MD, 1981.

[68] M. F. NEUTS. The caudal characteristic curve of queues. *Advances in Applied Probability*, 18:221–254, 1986.

[69] M. F. NEUTS. *Structured Stochastic Matrices of M/G/1-type and their Applications.* Marcel Dekker, New York, NY, 1989.

[70] B. F. NIELSEN. Modeling long-range dependent and heavy-tailed phenomena by matrix analytic methods. In *Advances in Matrix Analytic Methods for Stochastic Models*, G. Latouche and P. Taylor, editors, pages 265–278, NJ, 2000. Notable Publications Inc.

[71] B. F. NIELSEN. Modelling long-range dependent and heavy-tailed phenomena by matrix analytic methods. In *Advances in Algorithmic Methods for Stochastic Models*, G. Latouche and P. Taylor, editors, pages 265–278. Notable Publications, 2000.

[72] M. OLSSON. The EMpht-programme. Technical Report http://www.math.lth.se/matstat/staff/asmus/pspapers.htm, Department of Mathematics, Chalmers University of Technology, June 1998.

[73] V. S. PAI, M. ARON, G. BANGA, M. SVENDSEN, P. DRUSCHEL, W. ZWAENEPOEL, AND E. NAHUM. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, (ASPLOS-VIII)*, San Jose, CA, October 1998.

[74] L. R. RABINER. A tutorial on hidden Markov models and selected applications in speach recognition. *IEEE Transactions on Speech and Audio Processing*, 77(2):257–286, February 1989.

[75] V. RAMASWAMI. A stable recursion for the steady state vector in Markov chains of M/G/1-type. *Commun. Statist. Stochastic Models*, 4:183–189, 1988.

[76] V. RAMASWAMI. A duality theorem for the matrix paradigms in queueing theory. *Commun. Statist. Stochastic Models*, 6(1):151–161, 1990.

[77] V. RAMASWAMI AND G. LATOUCHE. A general class of Markov processes with explicit matrix-geometric solutions. *OR Spektrum*, 8:209–218, August 1986.

[78] V. RAMASWAMI AND J. L. WANG. A hybrid analysis/simulation for ATM performance with application to quality-of-service of CBR traffic. *Telecommunication Systems*, 5:25–48, 1996.

[79] A. RISKA, V. DIEV, AND E. SMIRNI. Efficient fitting of long-tailed data sets into hyperexponential distributions. In *Proceedings of IEEE Globecom Conference, Internet Performance Symposium*, Taipei, Taiwan, November 2002. IEEE Catalog Number: 02CH3798C.

[80] A. RISKA, V. DIEV, AND E. SMIRNI. Efficient fitting of long-tailed data sets into phase-type distributions. In *Proceedings of the Fourth Workshop on Mathematical Performancec Modeling and Analysis (MAMA2002)*, Marina Del Rey, CA, June 2002.

[81] A. RISKA AND E. SMIRNI. Exact aggregate solutions for M/G/1-type Markov processes. In *Proceedings of ACM SIGMETRICS Conference*, pages 86–96, Marina del Rey, CA, June 2002.

[82] A. RISKA AND E. SMIRNI. MAMSolver: a matrix-analytic methods tools. In *Proceedings of TOOLS 2002: LNCS 2324*, T. Field, P. Harrison, J. Bradley, and U. Harder, editors, pages 205–211. Springer-Verlag, 2002.

[83] A. RISKA AND E. SMIRNI. M/G/1-type Markov processes: A tutorial. In *Performance Evaluation of Complex Computer Systems: Techniques and Tools, LNCS 2459*, M. C. Calzarossa and S. Tucci, editors, pages 36–63. Springer-Verlag, 2002.

[84] A. RISKA, E. SMIRNI, AND G. CIARDO. Analytic modeling of load balancing policies with heavy-tailed distributions. In *Proceedings of the Second International Workshop on Software and Performance (WOSP'00)*, pages 147–157, Ottawa, Canada, September 2000.

[85] A. RISKA, E. SMIRNI, AND G. CIARDO. Exact analysis of a class of GI/G/1-type performability models. *IEEE Transactions on Reliability*, (forthcoming).

[86] A. RISKA, M. SQUILLANTE, S. YU, Z. LIU, AND L. ZHANG. Matrix-analytic analysis of a MAP/PH/1 queue fitted to web server data. In *Matrix-Analytic Methods: Theory and Applications*, G. Latouche and P. Taylor, editors, pages 333–356. World Scientific, 2002.

[87] A. RISKA, W. SUN, E. SMIRNI, AND G. CIARDO. ADAPTLOAD: effective balancing in clustered web servers under transient load conditions. In *Proceedings of the* 22th *International Conference on Distributed Computer Systems. (ICDCS'02)*, pages 103–111, Vienna, Austria, July 2002.

[88] S.M. ROSS. *Introduction to Probability Models. sixth edition.* Academic Press, San Diego, CA, 1997.

[89] M. ROSSITER. *Characterizing a random point process by a switched poisson process.* Monash University, Melbourne, 1989. PhD thesis.

[90] E. ROSTI, E. SMIRNI, G. SERAZZI, L.W DOWDY, AND K.C. SEVCIK. Processor saving scheduling policies for multiprocessor systems. *IEEE Transactions on Computers*, 47(2):178–189, February 1998.

[91] M. J. RUSELL AND R. K. MOORE. Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proceedings of ICASSP '85*, pages 5–8, 1985.

[92] T. RYDEN. Parameter estimation for Markov modulated poisson processes. *Commun. Statist. Stochastic Models*, 10(4):795–829, 1994.

[93] H. SCHELLHAAS. On Ramaswami's algorithm for the computation of the steady state vector in Markov chains of M/G/1 type. *Commun. Statist. -Stochastic Models*, 6:541–550, 1990.

[94] E. SENETA. *Non-Negative Matrices and Markov Chains. second edition.* Springer Verlag, New York. 1981.

[95] K. SIGMAN. A primer on heavy-tailed distributions. *Queueing Systems*, 33:261–275, 1999.

[96] H. A. SIMON AND A. ANDO. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[97] B. SIN AND J. H. KIM. Nonstationary hidden Markov model. *Signal Processing*, 46:31–46, 1995.

[98] C. SINGH AND R. BILLINTON. *System Reliability Modeling and Evaluation.* Hutchinson, 1977.

[99] E. SMIRNI. E. ROSTI. L.W DOWDY. AND G. SERAZZI. A methodology for evaluation of multiprocessor non-preemptive allocation policies. *Journal of Systems Architecture*. 44:703–721, 1998.

[100] M. S. SQUILLANTE. Matrix-analytic methods in stochastic parallel-server scheduling models. In *Advances in Matrix-Analytic Methods for Stochastic Models: Lecture Notes in Pure and Applied Mathematics*, S. R. Chakravarthy and A. S. Alfa, editors, NJ, 1998. Notable Publications.

[101] M. S. SQUILLANTE. Matrix-analytic methods: Applications. results and software tools. In *Advances in Matrix-Analytic Methods for Stochastic Models*. G. Latouche and P. Taylor. editors, NJ, 2000. Notable Publications.

[102] M. S. SQUILLANTE AND E.D. LAZOWSKA. Using processor cache affinity information in shared memory multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):131–143, February 1993.

[103] M. S. SQUILLANTE AND R.D. NELSON. Analysis of task migration in shared-memory multiprocessor systems. In *Proceedings of the ACM SIGMETRICS Conference*, pages 143–155, San Diego, CA, May 1991.

[104] M. S. SQUILLANTE, F. WANG, AND M. PAPAEFTHYMIOU. Stochastic analysis of gang scheduling in parallel and distributed systems. *Performance Evaluation*, 27-28:273–296, 1996.

[105] 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.

[106] M. S. Squillante, Y. Zhang. A. Sivasubramaniam. N. Gautam, H. Franke, and J. Moreira. Modeling and analysis of dynamic coscheduling in parallel and distributed environments. In *Proceedings of the ACM SIGMETRICS Conference*, pages 43–54, Marina del Rey, CA, May 2002.

[107] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, NJ, 1994.

[108] S.V.Vaseghi. State duration modeling in hidden Markov models. *Signal Processing*, 41:31–41, 1995.

[109] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[110] C. K. U, Y. K. Park, and O. W. Kwon. Modeling acoustic transitions in speech by modified hidden Markov models with state duration and state duration-dependent observation probabilities. *IEEE Transactions on Speech and Audio Processing*, 4(5):389–392. September 1996.

[111] C. Xia, Z. Liu, M. S. Squillante, L. Zhang, and N. Malouch. Traffic modeling and performance analysis of commercial web sites. Technical report, IBM, 2001.

[112] X. Zhang, M. Barrientos, J. Chen, and M. Seltzer. Hacc: an architecture for cluster-based web server. In *Proceedings of the 3rd USENIX Windows NT Symposium*, pages 155–164, Seattle, WA, July 1999.

[113] S. Zhou. A trace driven simulation study of load balancing. *IEEE Transactions on Software Engineering*, 14(9):1327–1341, 1988.

[114] S. Zhou, J. Wang, X. Zheng, and P. Delisle. Utopia: A load-sharing facility for large heterogenous distributed computing systems. *Software-Practice and Experience*, 23(2):1305–1336, 1993.

[115] H. Zhu, H. Tang, and T. Yang. Demand-driven service differentiation for cluster-based network servers. In *Proceedings of INFOCOMM'2001 Conference*, Anchorage, Alaska, April 2001.

VITA

Alma Riska

Born in Saranda, Albania on January 12th 1972. Awarded the third place on the National Olympiad in Mathematics for High School students in 1989. Graduated from "Hasan Tahsini" gymnasium honored with gold medal for excellent performance. Completed her diploma thesis at the Computer Center of University of Siegen, Germany, in the framework of the TEMPUS European Union project. Graduated with a Diploma from the School of Natural Sciences of Tirana University, on January 1996. From 1996 to summer of 1998, she continued to work at Computer Center of University of Siegen as part of the team responsible for installation and management of the university computer network. In August 1998, she started her graduate studies at the Department of Computer Science at College of William and Mary and is working ever since toward her PhD degree. While at William and Mary, she got involved on analytic modeling and methodologies for performance analysis of computer systems, work that she has published and presented in various professional journals and conferences.