Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

LARGE MARKOV MODELS FOR COMPUTER PERFORMANCE AND RELIABILITY ANALYSIS: EFFICIENT METHODS FOR DETERMINATION OF ERROR BOUNDS

C. S. Lui

October 1991
CSD-910070

UNIVERSITY OF CALIFORNIA

Los Angeles

LARGE MARKOV MODELS FOR COMPUTER PERFORMANCE AND RELIABILITY ANALYSIS: EFFICIENT METHODS FOR DETERMINATION OF ERROR BOUNDS

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Chi Shing Lui

1991

The dissertation of Chi Shing Lui is approved.

John W. Mamer

John W. Mamer

Steven Lippman

Steve A. Lippman

Mario Gerla

Mario Gerla

Leonard Kleinrock

Leonard Kleinrock

Richard R. Muntz, Committee Chair

University of California, Los Angeles

1991

I dedicate this dissertation to my wife, Irene, to whom I owe everything. Without her love, support, and joyful companionship, I would have never been able to complete my graduate studies.

Table of Contents

1 INTRODUCTION 1

1.1 Statement of the Problem 1

1.2 Contribution of this Dissertation 3

2 RELATED WORK 6

2.1 Solving Large Markov Model 6

2.2 Reliability/dependability 15

2.3 Joining the Shortest Queue Routing Policy 19

3 AVAILABILITY BOUNDING METHODOLOGY 22

3.1 Introduction 23

3.2 Markov Model and Assumptions 25

3.3 One-step Bounding Algorithm 27

3.4 Multi-step Bounding Algorithm. 35

3.5 Bound Spread Reduction. 45

3.5.1 Bound spread reduction for the clone states. 46

3.5.2 Bound spread reduction for the detail states. 50

3.5.3 Bound spread reduction algorithm. 52

3.6 Decision criteria for backward iteration or forward generation . . . 56

3.7 Partial state generation 61

3.8 Example. 70

3.9 Conclusions. 76

4 GENERALIZATION FOR AVAILABILITY BOUNDING METHODOLOGY 77

4.1 Problem Definition 78

4.2 Approach 79

4.3 Example 88

5 ALGORITHMIC APPROACH TO BOUNDING THE MEAN RESPONSE TIME OF A MINIMUM EXPECTED DELAY ROUTING 92

5.1 Introduction 92

5.2 Minimum Expected Delay Routing Model 95

5.3 Upper Bound Model 98

5.3.1 Proof of upper bound mean response time 101

5.3.2 Computational algorithm for solving the upper bound model 105

5.4 Lower Bound Model 110

5.4.1 Proof of lower bound mean response time 111

5.4.2 Computational algorithm for solving the lower bound model 116

5.5 Further State Space Reduction 119

5.6 Numerical Examples 120

5.7 Conclusions 123

6 CONCLUSIONS 124

A Proof for Theorem 3.4 128

B Proof for Theorem 3.5 132

References 135

List of Figures

2.1 Combinatoric Analysis. 15

2.2 Fault tree representation. 16

3.1 Transition matrix G. 27

3.2 Relationship of G and G'. 31

3.3 Form of transition matrix after aggregation. 33

3.4 Form of rate matrix after bounding transition rates. 34

3.5 Rate matrix G. Initial matrix. 38

3.6 Rate matrix G_1. Introduction of clone states. 38

3.7 Rate matrix G_2. After exact aggregation of the states in $\mathcal{D}'$. . . . 39

3.8 Rate matrix G_3. Modified rates from state d'. 40

3.9 Rate matrix G_4. Exact aggregation of states in $\mathcal{C}_i$ and $\mathcal{A}_i$. 41

3.10 Rate matrix G_5. Replacement of transition rates with bounds. . . 42

3.11 Transition matrix G_1. 65

3.12 Transition matrix G_2. 66

3.13 Transition matrix G_3. 67

3.14 Transition matrix G_4. 68

3.15 Transition matrix G_5. 69

3.16 A fault-tolerant heterogeneous distributed database system. . . . 71

4.1 rate matrix $\hat{G}_0'$ 83

4.2 rate matrix $\hat{G}_0''$ 84

4.3 A highly fault-tolerant distributed system with module repair. . 89

5.1 Transition probability matrix for upper bound model. 106

5.2 Aggregate Chain for upper bound model 109

5.3 Transition rate matrix for M. 112

5.4 Transition rate matrix for M_1. 113

5.5 Transition rate matrix for M_2. 114

5.6 Transition rate matrix for M_3. 115

5.7 Transition rate matrix for lower bound model. 117

5.8 Aggregate Chain for lower bound model 118

List of Tables

3.1 Failure and repair rates(per hour). 73

3.2 Upper and lower bounds on steady state availability of the database system. 74

3.3 Contribution by clones states and detail states for step 3 74

3.4 Contribution by clones states and detail states for step 5 75

3.5 Illustration for decision algorithm 75

4.1 Failure and repair rates(per hour). 90

4.2 Availability bounds on highly fault-tolerant distributed system. . 91

5.1 Homogeneous servers system 121

5.2 Heterogeneous servers system 122

5.3 Computational Cost vs. Accuracy 123

Acknowledgments

I would like to express my appreciation to the members of the committee, Professor John W. Mamer, Steve A. Lippman, Mario Gerla, Leonard Kleinrock and Richard Muntz. I am deeply grateful to Professor Richard Muntz, my friend and advisor, who taught me not to work at a level of mediocrity, but always strive for a level of excellence. Through his dedication and sincerity, I learned what the meaning of research is.

Special thanks to my friends in the performance modeling group, Bill Cheng, Leana Golubchik, Steve Berson, and Gary Rozenblat; they provided a lot of fun and exciting discussions during the course of this dissertation. To my brothers and sisters in the Chinese Bible Church, thanks for their encouragement through out the course of my graduate studies.

I would like to thank the Computer Science Department staff for their invaluable help. In particular, I would like to mention: Verra Morgan, Rosie Murphy, and Doris Sublette.

To my parents, who brought me up in love and instilled in me the courage and determination to hang on during the difficult times, I shall be forever grateful.

VITA

1962	Born, Hong Kong
1984	B.S. Electrical Engineering. Illinois Institute of Technology.
1984-1986	Teaching Assistant, Electrical Engineering Department, Illinois Institute of Technology.
1986	M.S. Electrical Engineering. Illinois Institute of Technology
1986-1988	Teaching Associate, Computer Science Department, UCLA.
1988-1991	Research Assistant, Computer Science Department, UCLA.

PUBLICATIONS

John C.S. Lui, R.R. Muntz, "Evaluating Bounds on Steady State Availability of Repairable Systems from Markov Models", First International Conference on the Numerical Solution of the Markov Chains, January, 1990.

R.R. Muntz, John C.S. Lui, "Performance Analysis of Disk Arrays Under Failure", 16^{th} International Conference on Very Large Data Bases, 1990.

John C.S. Lui, R.R. Muntz, "A Bounding Methodology for Computing Steady State Availability of Repairable Computer Systems". Submitted to JACM for publication

Leana Golubchik, John C.S. Lui, Richard R. Muntz, "Chained Declustering: Load Balancing and Robustness to Skew and Failures", UCLA Technical Report CSD-910053. Submitted to 2^{nd} International Workshop on Research Issues on Data Engineering: Transaction and Query Processing (RIDE-TQP) for publication.

John C.S. Lui, R.R. Muntz, "Algorithmic Approach to Bounding the Mean Response Time of a Minimum Expected Delay Routing Systems". UCLA Technical Report CSD-910064. Submitted to ACM SIGMETRICS for publication

Abstract of the Dissertation

LARGE MARKOV MODELS FOR COMPUTER PERFORMANCE AND RELIABILITY ANALYSIS: EFFICIENT METHODS FOR DETERMINATION OF ERROR BOUNDS

by

Lui Chi Shing

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1991

Professor Richard R. Muntz, Chair

With the advance of computing technologies, we are able to build larger and more complex computer systems. To ensure the performance as well as the reliability of these systems, performance evaluation becomes a crucial component during the design process. Markov models are widely used by performance analysts because of their generality to represent the complex interactions between components in real life systems. Most often the Markov model we want to analyze has irregular structure and a closed-form solution is not known and to solve the Markov model, we resort to numerical solution techniques. The most pervasive limitation of numerical techniques is the inability to handle the very large state spaces which are often required to represent realistic models.

In this dissertation, we develop a methodology for analyzing large availability Markov models for highly fault-tolerant computer systems. The state space cardinality of this kind of availability models is very large and it is prohibitive to generate all the transition rate matrix. The methodology we present can (i) bound the system steady state availability and at the same time, (ii) drastically reduce the state space of the model that must be solved. The bounding methodology is also iterative and generates part of the transition rate matrix at each step. At each step tighter errors bounds on the system availability are obtained.

We also develop a methodology to analyze a load balancing algorithm which used minimum expected delay as routing policy. This kind of policy can be view as a generalization of join the shortest queue routing policy. The state space cardinality of the Markov model is infinite and no closed-form solution exists in general. We present an numerical algorithm which can (i) bound the expected response time of job (and expected number of jobs in the system) and, (ii) reduce the state space of the model that need to be generated.

CHAPTER 1

INTRODUCTION

In this chapter, we describe the problem we are studying and state the contribution of this dissertation.

1.1 Statement of the Problem

In recent years, the computer industry has made tremendous progress in providing processing power, computer network technologies, storage technologies, etc. Research results in fields like algorithms, database theory and distributed and parallel processing are also widely applied. These results enable computer system designers to build larger and more complex systems. One of the requirements of these complex systems is that they have a satisfactory performance under nominal conditions. Some of the common performance measures that need to be optimized are expected response time, system throughput, ... etc. In order to achieve an acceptable performance in today's highly parallel systems, system resources such as processors, disks or memory buffers, should be evenly utilized. One way to evenly utilize the system resources is by using some form of load balancing algorithms. In addition to performance under nominal conditions, another important requirement for these complex computer systems is that they be highly fault-tolerant. These systems must not only be able to perform at full capacity when all components are fully operational but also, when faults occur (either due to software, hardware or external source), there is often a requirement that they still function correctly and at some guaranteed level of degraded mode performance.

Since complex computer systems are often expensive and/or used in critical applications (e.g., air traffic control, banking transaction, etc), it is increasingly important for computer designers to be able to predict the performance and dependability properties of the system during the design phase to verify that the system will meet specific fault-tolerant and performance requirements.

Simulation is one method of analyzing the system performance and dependability. There are several inherent problems of using simulation in analyzing computer systems. First, it is very time consuming and expensive to obtain the desire result. Secondly, it is difficult to check the correctness of the simulation. The problem is aggravated if we use simulation for dependability analysis for complex computer systems because these systems are built to be highly fault-tolerant. System failure becomes a rare event phenomenon, therefore, to get tight confidence intervals of the system reliability/availability measures via simulation can require excessively long simulation time. Recent work on rare simulation techniques [CG87, SHG88, NNH90, GSN89] holds promise for broadening the applicability of simulation in availability analysis.

Another approach to system performance and dependability analysis is to use Markov models. Most of the dependability models or load balancing models have irregular structure and closed form solutions are extremely difficult to obtain. Therefore, numerical solution techniques are most often used to solve the Markov model. The most prohibitive limitation with using numerical solution techniques is that for realistic systems the model often has an unmanageably large state space and it quickly becomes impractical to even generate the entire transition rate matrix for the system model.

The problem we study in this dissertation is whether one can use approximation techniques to obtain bounds on performance measures of Markov models that have too large (or even infinite) state space to be solved numerically. Specifically, we explore how one can trade exact performance analysis for bounds and in exchange obtain a reduction in computational cost. In this dissertation, we study Markov models for dependability and performance analysis. Dependability analysis via Markov models usually has finite but very large state space. For performance analysis using Markov models, we study load balancing algorithms that are aimed at evenly utilizing system resources. Specifically, we study the classic *join the shortest queue* load balancing algorithm. This algorithm is appealing not only due to it's simplicity in implementation, but theoretically difficult to analyze. Since the arrival process is state dependent and no close-form solution exits in general. Since each servers has an infinite capacity queue, the state space cardinality of the Markov model is infinite.

1.2 Contribution of this Dissertation

In this section, we outline the contributions of this dissertation. In chapter 3, we propose a methodology to evaluate computer system dependability via Markov models. There are two appealing properties in the methodology we develop, namely, (i) bounds on the system steady state availability are obtained, (ii) the state space of the model that must be solved is drastically reduced. The bounding algorithm is iterative and generates part of the transition matrix at each step.

At each step tighter bounds on system availability are obtained. The algorithm also allows the size of the submodel to be solved at each step be chosen to accommodate memory limitations. This general bounding methodology provides an efficient way to evaluate reliability models with very large state spaces without ever generating the entire transition rate matrix. We emphasize that the method provides error bounds for availability and not just an approximation.

The dependability bounding methodology proposed in Chapter 3 assumes that the original Markov process has an upper block Hessenberg form. In chapter 4, we generalize the bounding methodology to a Markov process with a general transition structure.

In chapter 5 we study a load balancing algorithm in which the routing decision is based on minimum expected delay for the arriving job. This type of load balancing algorithm is a generalization of the shortest queue routing load balancing algorithm. We present an algorithmic approach to bounding the mean response time of a multi-server system in which the minimum expected delay routing policy is used, i.e. an arriving job will join the queue which has the minimal expected value of remaining workload. We assume the queueing system to have K servers, each with an infinite capacity queue. The arrival process is Poisson with parameter λ and the service time distribution on server i is exponentially distributed with mean $1/\mu_i, 1 \leq i \leq K$. Without loss of generality, we assume $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$. There is a rich literature in which the performance of the shortest queue routing load balancing algorithm is studied, but none of the previous work treats more than two servers and simultaneously, provides error bounds. The major contribution of our computation algorithm is that it (1) allows more than $K \geq 2$ servers, (2) allows heterogeneous servers, (3) includes scheduling based on queue length and service rate (thus, a generalization of joining the shortest queue) and (4) provides error bounds. This bounding methodology also allows one to tradeoff accuracy and computational cost as will be demonstrated.

CHAPTER 2

RELATED WORK

In this chapter, we give an overview of previous research related to this dissertation. We divide the related work into three parts. The first part of this chapter concerns work related to solving large Markov model and bounding of performance measure based on the Markov model. The second part discusses related work on dependability analysis. The last part concerns performance analysis of the join-the-shortest-queue load balancing algorithm.

2.1 Solving Large Markov Model

To analyze the performance of a system, we often represent the system by a Markov model and attempt to solve the model. Since closed-form solutions are available only in a limited number of cases, such as simple queueing systems, non-blocking Jackson-type queueing network, ..., etc. Most often the analysis approach for solving the non-closed-form Markov models is by numerical or approximate computation.

Some Markov models have special transition structures or properties which make them easily analyze. One of these properties is *reversibility*, which was first studied by Kingman [Kin69]. Intuitively, a Markov process $X(t)$ which satisfied reversibility is one in which the direction of time has no effect on the statistics of the process. Therefore, $X(t)$ and $X(-t)$ have identical statistical properties. A necessary and sufficient condition for a Markov process to be reversible is that it satisfies the detailed balanced equations. Formally, a stationary Markov process with state space $\mathcal{S}$ is reversible if and only if the following conditions are satisfied:

$$\pi(i)q(i,j) = \pi(j)q(j,i) \qquad i,j \in \mathcal{S} \tag{2.1}$$

$$\sum_{i \in \mathcal{S}} \pi(i) = 1 \tag{2.2}$$

where $\pi(i)$ is the steady state probability of state i and $q(i,j)$ is the transition rate from state i to state j. Finding the closed-form solution for a reversible process is simplified because a reversible process satisfy detailed balanced equations. The simplest examples of reversible Markov process is a birth-death process.

There are some Markov processes which have the matrix-geometric form. Neuts [Neu81] developed a body of elegant results in solving numerically this kind of Markov processes. The transition rate matrix of this kind of Markov process can be characterized by an infinite and repetitive structure in terms of finite vectors of states. In general, the block generator matrix of this kind of irreducible Markov process can has the form:

$$\begin{vmatrix} B_0 & A_0 & 0 & 0 & 0 & \cdots \\ B_1 & A_1 & A_0 & 0 & 0 & \cdots \\ B_2 & A_2 & A_1 & A_0 & 0 & \cdots \\ B_3 & A_3 & A_2 & A_1 & A_0 & \cdots \\ B_4 & A_4 & A_3 & A_2 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{vmatrix}$$

with vector $\mathbf{x} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots]$ being the steady state probability vector. Solution to steady state probability vector $\mathbf{x}$ is reduced to solving the following:

$$\mathbf{x}_0 \sum_{k=0}^{\infty} \mathbf{R}^k \mathbf{B}_k = \mathbf{0} \tag{2.3}$$

$$\mathbf{x}_k = \mathbf{x}_0 \mathbf{R}^k \qquad k \geq 1 \tag{2.4}$$

$$\mathbf{x}_0 (\mathbf{I} - \mathbf{R})^{-1} \underline{e} = \mathbf{1} \tag{2.5}$$

where $\mathbf{R}$ can be solved iterative by the following procedure.:

$$\mathbf{R}(0) = \mathbf{0} \tag{2.6}$$

$$\mathbf{R}(n+1) = -\sum_{v=1, v\neq 1}^{\infty} \mathbf{R}^v \mathbf{A}_v \mathbf{A}_1^{-1} \tag{2.7}$$

It can be shown that $\lim_{n\to\infty} \mathbf{R}(n) = \mathbf{R}$. The Markov process is stable if and only if:

$$\pi \mathbf{A}_0 \underline{e} < \sum_{k=2}^{\infty} (k-1) \pi \mathbf{A}_k \underline{e} \tag{2.8}$$

where:

$$\pi \sum_{l=0}^{\infty} \mathbf{A}_l = \mathbf{0} \qquad and \tag{2.9}$$

$$\pi \underline{e} = 1 \tag{2.10}$$

A simple example of Markov process which has matrix-geometric form is $M/G/1$ queueing system.

For some Markov processes, although their state space cardinality is large, they exhibit an *nearly completely decomposable* (NCD) form. Roughly speaking, an irreducible Markov process is nearly completely decomposable if the interactions between groups of states are not comparable with the interactions within the groups. To illustrate the idea of NCD, let **P** be the stochastic matrix for an irreducible Markov chain with dimension n and **x** be the steady state probability vector. We said **P** is NCD if there exits a completely decomposable stochastic matrix:

$$\mathbf{P}^* = \begin{bmatrix} \mathbf{P}_1^* & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2^* & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_{n-1}^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P}_N^* \end{bmatrix}$$

where $\mathbf{P}_i^*$ is a square stochastic matrix. Let $|\mathbf{P}_i^*|$ be the order of $\mathbf{P}_i^*$, then $n = \sum_{i=1}^{N} |\mathbf{P}_i|$. The relationship between the original stochastic matrix P and P^* is:

$$\mathbf{P} = \mathbf{P}^* + \epsilon\, \mathbf{C} \tag{2.11}$$

where $\mathbf{C}$ is a square matrix with order n and ϵ is a real positive number which is small compare to the elements in $\mathbf{P}^*$. Simon and Ado [SA61] first showed that (1) in the short-run dynamics, a local equilibrium is reached by the strong interactions within each subsystem almost independently of the other subsystems. (2) In the long-run dynamics, the weak interactions among groups make themselves felt and the whole system moves towards a global equilibrium values attained by the state variables of each subsystem at the end of the short-run dynamics period. These properties indicate that the short-run equilibrium statistics can be approximately analyzed without consideration of other subsystems. Once the local equilibrium of each subsystems is obtained, they can be represented by an aggregated variable and the long-run dynamics of the whole system can be analyzed as a set of interactions between the newly formed aggregates.

It is noteworthy to mention the technique of aggregation to reduce the state space cardinality of the Markov process we want to analyze because in some situations, we may be only interested the steady state performance of the system at a "macro" level. In [Cou77], Courtois illustrated the existence of exact aggregation of Markov model. To illustrate the idea of exact aggregation, let us consider an irreducible Markov process with state space $\mathcal{S}$. We partition the state space into two disjoint sets, namely, $\mathcal{S}_1$ and $\mathcal{S}_2$ (partitioning the state space into two sets of disjoint sets is only for ease of exposition). The transition rate matrix of the Markov process is:

$$\begin{bmatrix} Q_{1,1} & Q_{1,2} \\ \\ Q_{2,1} & Q_{2,2} \end{bmatrix}$$

where $Q_{i,j}$ is the transition rate matrix from $\mathcal{S}_i$ to $\mathcal{S}_j$. Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$, be the steady state probability vector for the Markov process. Courtois showed that we can aggregate all states in $\mathcal{S}_i$ into a single state, s_i, $i = 1, 2$. The aggregated process has the rate matrix :

$$\begin{bmatrix} \bullet & q_{1,2} \\ q_{2,1} & \bullet \end{bmatrix}$$

where:

$$q_{i,j} \quad = \quad (\mathbf{x}_i \; \underline{e})^{-1} \mathbf{x}_i \; Q_{i,j} \; \underline{e} \tag{2.12}$$

Unfortunately, exact aggregation often requires the knowledge of conditional state probabilities for the set of states we want to aggregate, which in turn requires the steady state probability vector of the original model. Yet the importance of the result is that we can at least conceptually apply exact aggregation to the given Markov model, and therefore drastically reduce the state space cardinality of the problem. By properly applying the transition rates between aggregates, we get obtain an approximate result. One important note is that the aggregated process, in general, is not a Markov process. Therefore, we cannot assume the distribution of time staying in an aggregated state is exponential.

Kemeny and Snell [KS60] studied under what conditions an aggregated process is still Markovian, and which they coined the name, *lumpability conditions*. They establish the necessary and sufficient conditions that must be satisfied by the lumping process so that Markovian properties are preserved. If the original Markov model satisfies the lumping criteria, a lumped process can be obtained and the state space of the original problem is greatly reduced. A necessary and sufficient condition for a Markov process to be lumpable with respect to a partition $\{\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_N\}$, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ is that for every pair of sets $\mathcal{S}_i$ and $\mathcal{S}_j$,

$r_{k,\mathcal{S}_j}$ have the same values for every state $k \in \mathcal{S}_i$, where:

$$r_{k,\mathcal{S}_j} \;=\; \sum_{l \in \mathcal{S}_j} q_{k,l} \qquad\qquad for\ k \in \mathcal{S}_i \tag{2.13}$$

It is obvious that the lumpability is a very strong condition to satisfied.

One of the most common approximation method to solve Markov model with large or infinite state space is by state space truncation. One of the most common approaches to perform truncation is simply by cutting off the transition rates outside the set of states we want to analyze. That is given rate rate matrix:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ \\ Q_{21} & Q_{22} \end{bmatrix}$$

with $|Q_{11}| = n$ and we want to truncate all transition outside submatrix Q_{11}. We form a new rate matrix Q^* where:

$$Q^* \;=\; Q_{11} \;+\; diag\,(Q_{12}\ \underline{e}) \tag{2.14}$$

where $diag\,(Q_{12}\ \underline{e})$ is a diagonal matrix with it's i^{th} diagonal element being the i^{th} element of column vector $(Q_{12}\ \underline{e})$. Various truncation approaches were presented in [GS87] like augmenting the external row sum to the i^{th} column of the matrix Q_{11}. Although state space truncation is quite commonly apply in practice, there are some practical and theoretical problems in applying the truncation. In practice, we not only have to make sure that the resulting Markov process after truncation is irreducible and from the theoretical point of view, errors introduced by truncation is also very difficult to quantify. Convergence proofs as the truncation size tends to infinity has been studied by Seneta [Sen80]. Seneta also provide a simple and robust error bounds but these bounds do not secure an order of accuracy.

If the original Markov model is difficult to analyze. One approach to circumvent the problem is to analyze another Markov model by perturbing the transition structure of the original model. Schweitzer [Sch68] studied this problem and quantified the errors introduced in terms of the steady state probability vector and the fundamental matrix of the perturbed Model. It is important to point out that to obtain the fundamental matrix of a Markov model involves performing matrix inverse and it is computationally expensive.

In some cases, system designers are interested in a performance measures which depend primarily on how the system behaves in a certain restricted subset of states. In [CS86], Courtois and Semal illustrated the idea of obtaining bounds on conditional steady-state probabilities in a large Markov model. Given a stochastic matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1N} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_{N1} & \mathbf{P}_{N2} & \cdots & \mathbf{P}_{NN} \end{bmatrix}$$

Suppose we want to bound the conditional probabilities in the first block. Let $\mathbf{v}_i$, $i = 1, \ldots, N$, be the conditional state probability vector over the states within the i^{th} block. If $\mathbf{P}_{11}$ is the only information provided, then let:

$$\mathbf{Z}(0) = \mathbf{A}^{-1}(0)\,(\mathbf{I} - \mathbf{P}_{11})^{-1} \qquad (2.15)$$

where $\mathbf{A}^{-1}(0)$ is a diagonal matrix which normalizes to one the row sums of the matrix $(\mathbf{I}-\mathbf{P}_{11})^{-1}$, then $\mathbf{v}_1$ can be bounded by:

$$\min_i[\mathbf{Z}(0)]_{ik} \quad \leq \quad [\mathbf{v}_1]_k \quad \leq \quad \max_i[\mathbf{Z}(0)]_{ik} \tag{2.16}$$

If $\mathbf{P}_{j1}$, $j=1,\ldots,N$ are also available. Let

$$\mathbf{Z}_j(1) \quad = \quad \mathbf{A}_j^{-1}(1)\ \mathbf{P}_{j1}\,(\mathbf{I}-\mathbf{P}_{11})^{-1} \qquad\qquad j=1,2,\ldots,N \tag{2.17}$$

and $\mathbf{v}_1$ can be bounded by:

$$\min_{i,j}[\mathbf{Z_j}(1)]_{ik} \quad \leq \quad [\mathbf{v}_1]_k \quad \leq \quad \max_{i,j}[\mathbf{Z}_j(1)]_{ik} \tag{2.18}$$

They also proved that these bounds are the tightest ones that can be obtained given only knowledge of the submatrix $\mathbf{P}_{11}$ (and of the second case, $\mathbf{P}_{j1}$ for $j=1,\ldots,N$) [CS84, CS85].

In [Dij90], Van Dijk illustrated that the use of Markov reward proof techniques to compare different Markov models and thereby obtain performance bounds. Van Dijk used this proof technique to show that by modifying the original system (which has no close-form solution) to obtain another system with closed-form solution, simple but loose bounds can be obtained. This technique has been applied in some queuing problems that have no close-form solution [Dij88b, Dij88a, Dij89a, Dij89b].

In this dissertation, we shall employ the techniques discussed above to reduce the state space of the problem. Specifically, we use the Markov reward proof techniques to show the model we modified indeed provides performance bounds. Unlikely the approach used by Van Dijk, we do not necessary modify the model to another model with closed-form solution. Rather, we modify the model such a way that we can efficiently computer the performance measures. Since the modification is not as drastic as [Dij88b, Dij88a, Dij89a, Dij89b], we obtain very tight performance bounds.

2.2 Reliability/dependability

Due to the growing dependence of society on computer systems, there is an increase interest in evaluating and predicting fault tolerance in computer systems. Dependability analysis has been an active area research for the past decade. Techniques like combinatorics [Tri82], fault tree analysis [BFS75], simulation [LB84, CG87] and Markov models [SG86, SG89, SG85a, GCS86, GLT86, HG87]. In recent years, tools [CF86, CDL81, GCS86, MA82, TDG84] have been built to aid system designers to evaluate system reliability.

Trivedi in [Tri82] used reliability block diagram to analyzed system reliability. If the system we want to analyze has a regular structure, e.g., serial or parallel configuration, and independent component failures and repairs, then it is easier to analyze the reliability of this system. For example, given the system in Figure 2.1, let R_i be the probability the component i is reliable. The system as a whole is consider reliable if there is a path from point A to point B. The probability of a reliable system, R_s, can be expressed as:

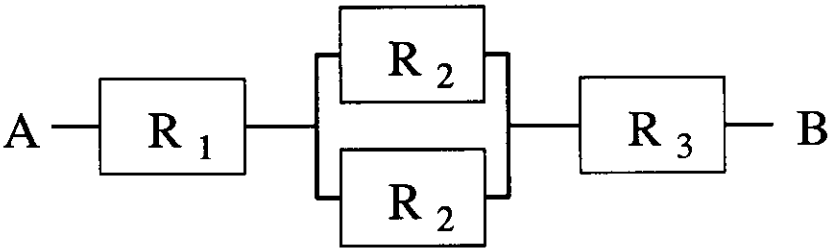

Figure 2.1: Combinatoric Analysis.

$$R_s \;=\; R_1\,\left[1-(1-R_2)^2\right]\,R_3 \tag{2.19}$$

But in real life situation, complex systems we want to analyze usually do not have any regular structure and components failures do not occur independently, that is, when one component fails, it will cause other components in the system to fail as well. Hence combinatoric analysis becomes no longer applies.

The Fault tree technique was originated by H.A. Watson of Bell Laboratories [Sin81] and techniques were further developed in [BFS75]. Fault tree uses fault tree symbols (ex: AND gate, OR gate, ...etc) to relate undesirable events in the system that eventually leads to system failure. A fault-tree presentation of system in Figure 2.1 is illustrated in Figure 2.2 is illustrated in: Where f_{R_i} represents the event that component i failed and f_s represents the system failure. Fault tree construction is a tedious process because it requires:

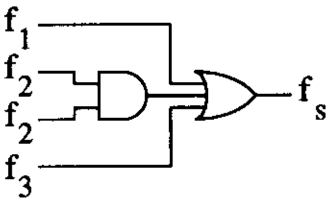

Figure 2.2: Fault tree representation.

1. specification of combination of failure events that lead to system failure.

2. application of a minimal cut set algorithm to remove redundant input events in the fault trees.

3. for large, complex systems, system designers may have to specify tens of thousands of fault tree symbols.

Many system analysts prefer simulation for system performance reliability evaluation. But naive simulation does not work on a highly fault-tolerant computer systems. Since computer systems are built to be fault-tolerant, system failures are so rare that it requires extremely long run of simulation to obtain samples of system failure states. One promising approach to overcome this type of problem is the concept of *importance sampling.* The basic idea is to perturb the probability measure such that system failures become more frequent. This is accomplished by making component failure events in the simulation occur with probabilities that are much higher than the actual values. In order to obtain the proper results, adjustments have to be made to the sample outputs to unbias the performance estimator. Lewis, Bohm [LB84] used the importance sampling technique to simulate the network reliability problem with independent components. Their approach does not apply when there is interactions or dependency between components. In [CG87], Conway and Goyal used a heuristic technique of importance sampling, known as *failure biasing*, to evaluate the dependability of highly fault-tolerant system. In [SHG88], failure biasing was used to estimate the mean time to failure of Markovian systems. Nicola [NNH90] also showed that by carefully selecting a heuristic for importance sampling, orders of magnitude reduction in simulation run-lengths can be obtained. In [GSN89], the optimal importance sampling distribution for dependability analysis of a three state example was derived. Yet, the study of failure biasing or importance sampling has been confined to the level of heuristics on very simple examples. Depending on the system which is being modeled, it may be difficult to determine an efficient importance sampling distribution.

Most research on dependability analysis represents the system in the form of a Markov model. This approach proceeds by enumeration of system states. The state transition rates are calculated, and the steady state reliability or dependability measures can be obtained by solving the Markov chain. Due to the complex interaction between system components, closed-form solution of system dependability is difficult if not impossible to obtain, therefore numerical methods are used in solving the Markov model. In [SG85a, GCS86, GLT86], the authors investigated several numerical techniques to evaluate dependability measures for Markovian systems with large state space. Some of the popular numerical methods are:

1. power method
2. successive overrelaxation (SOR)
3. direct method
4. iteration and aggregation
5. Unsymmetric Lanczos method

The SOR method is suggested in [SG85a] as the method of choice although the optimum relaxation parameter ω is difficult to obtain. Most of these numerical techniques are iterative in nature and the convergence condition or the convergence rate is difficult to guarantee.

Another limitation of direct application of numerical solution techniques is that a realistic system model often has an unmanageably large state space and it quickly becomes impractical to even generate the entire transition matrix.

All the dependability analysis discussed above either cannot handle large and complex systems or the confidence intervals of the dependability measures are costly to obtain. In this thesis, we develop a general bounding algorithm that can (i) drastically reduce the state space of the Markov model and, (ii) provides error bounds for the dependability measures. The bounding algorithm is iterative and generates part of the transition matrix at each step. At each step tighter bounds on system availability are obtained. The algorithm also allows the size of the submodel to be solved at each step be chosen to accommodate memory limitations. This general bounding methodology provides an efficient way to evaluate dependability models with very large state spaces without ever generating the entire transition rate matrix.

2.3 Joining the Shortest Queue Routing Policy

Multi-processor computing systems composed of many servers are now commonplace. In order to maximize the performance of the system, it is important to distribute the workload evenly among all processors. Joining the shortest queue is a natural way to balance the load in a multi-server system and thereby achieve better system performance, i.e. mean response time. In [Win77], Winston showed that given the arrival process is Poisson, join the shortest queue policy maximizes the discounted number of jobs which complete service by a certain time t. In [Web78], Weber generalized the result by relaxing the distribution of the arrival process. In [EVW80], Ephremides et al. showed that join the shortest queue policy is optimal with respect to delay in a system with two queues. It is interesting to point out that joining the shortest queue routing policy is both *socially* and *individually* optimal.

One of the major difficulties in analyzing this kind of routing discipline is the multidimensional nature of the state space, which is infinite in each of the dimensions. Most of the published results are limited to the case where the number of servers is equal to two and exponential interarrival and service times.

We give a brief review of the published literature on the shortest queue routing problem. Kingman [Kin61], and later Flatto and McKean [FM77] studied this problem with two servers via transform methods. They obtained an expression for mean number of jobs in the system expressed as an infinite sum which can be simplified under a heavy traffic assumption. Cohen and Boxma [CB83] treated a similar problem as a Reimann-Hilbert boundary problem and obtained a functional representation for the mean number of customers in the system. Conolly [Con84] studied the same model as in [FM77, Kin61] and proposed an approximation algorithm for evaluating equilibrium state probabilities via state truncation. Rao and Posner [RP87] proposed an approximation for a system with two servers and each server having different service rates (heterogeneous servers). An arriving job joins the server with smaller number of jobs (rather than joining the server with minimum expected delay). The analysis approach involves treating one of the queues as bounded so that the transition rate matrix for the modified system can be expressed in a matrix-geometric form [Neu81]. Grassman [Gra80] studied the same problem with two servers and solved for transient and steady state behavior. Halfin [Hal85] studied the two servers problem and used a linear programming technique to compute bounds on the mean number of customers in the system. Blanc [Bla87] studied the join shortest queue problem with arbitrary number of heterogeneous servers. He proposed an approximation method which is based on power series expansions and recursion which requires substantial computational effort. Nelson and Philips [NP89, NP90] proposed an approximation for mean response time of arbitrary number of servers. More importantly, the approximation allows general interarrival and service time distribution. None of the work cited above treats more than two servers and simultaneously provides error bounds.

In this dissertation, we develop a computational algorithm that (1) allows more than two servers, (2) allows heterogeneous servers, (3) includes scheduling based on queue length and service rate (thus, a generalization of joining the shortest queue) and (4) provides error bounds. This bounding methodology also allows one to tradeoff accuracy and computational cost.

CHAPTER 3

AVAILABILITY BOUNDING METHODOLOGY

One of the most important performance measures for computer system designers is system availability. Most often Markov models are used in representing systems for dependability/availability analysis. Due to complex interactions between components and complex repair policies, the Markov model often has an irregular structure and closed form solutions are extremely difficult to obtain. Another issue is that for a realistic system the model often has an unmanageably large state space and it quickly becomes impractical to even generate the entire transition rate matrix for the system model. In this chapter, we present a methodology that can (i) bound the system steady state availability and at the same time, (ii) drastically reduce the state space of the model that must be solved. The bounding algorithm is iterative and generates part of the transition matrix at each step. At each step tighter bounds on system availability are obtained. The algorithm also allows the size of the submodel to be solved at each step be chosen to accommodate memory limitations. This general bounding methodology provides an efficient way to evaluate dependability models with very large state spaces without ever generating the entire transition rate matrix.

3.1 Introduction

Computer systems are widely used in many applications (e.g., air traffic control and banking application) where dependability is crucial. System dependability analysis has long been an active area of research. Techniques such as combinatorics analysis, Markov or semi-Markov analysis [SG86, GT85, GLT86, HG87, Tri82], and simulation [LB84, CG87] have been used in dependability analysis. In recent years, tools [CF86, CDL81, GCS86, MA82, TDG84] have been built to aid system designers to evaluate and compare different architectures during the design process.

There are two major types of dependability measures that are of interest. The first type concerns transient measures, e.g, distribution of the number of times the system failed in a certain mode by time t. This type of measure is especially appropriate for mission oriented systems (e.g., spacecraft computers). The second type concerns steady state dependability measures such as steady state availability. These measures are appropriate for systems with lifetimes that are long enough to span many failure and repair cycles (e.g., database management systems or telephone systems). Methods to solve transient dependability measures of repairable computer systems have been reported in [SG86, SG89]. In this dissertation, we concentrate on steady state availability.

Markov models are most widely used in analyzing system dependability because of their generality as well as the ability to represent complex interactions among components (e.g., dependent failure rates, complex repair policy, etc). Because of these complex interactions between components, closed form solutions are extremely difficult if not impossible to obtain. Therefore, numerical solution techniques are often used in analyzing the Markov model. One of the most pervasive limitations of numerical techniques is the inability to handle large models. For models of realistic systems, the state space requirements often vastly exceed the memory and storage capacity of current (or future) systems [SG85b]. Although approximations (e.g., state space truncation) can be applied to solve the cardinality problem, the errors introduced are difficult to quantify. In this dissertation, we present a methodology to compute steady state availability that can drastically reduce the state space cardinality and at the same time, provide error bounds.

The results recently reported in [LM90, MSG89] provide methods for computing bounds on the steady state availability of repairable systems. In [MSG89], a 'one-step' algorithm was proposed which requires the user to make an a priori decision concerning the portion of the state transition matrix to be generated and the steady state availability bounds are computed based on this submodel. In [LM90], a 'multi-step' algorithm was proposed which allows a stepwise generation of submodels such that at each step, lower bound on the stationary state probabilities for those newly generated states are obtained. Hence, we can achieve progressively tighter steady state availability bounds. Each successive application of the algorithm in [LM90] can tighten the availability bounds but the spread between the bounds has a non-zero limiting value, i.e. the bounds cannot be made arbitrarily tight. In this dissertation, we present a general bounding methodology that augments the previous results by providing an iterative procedure to refine the bounds to arbitrary precision.

After any step in the iterative computation, we will have generated lower bounds for the stationary state probabilities for a subset of the states of the model. The lower bounds on the state probabilities are used to compute upper and lower bounds on steady state availability. The algorithm we present can then proceed in two ways: (i) another set states can be explored and lower bounds for their state probabilities can be computed or (ii) the lower bounds on the state probabilities which have been previously computed can be refined. We refer to the former as a *forward* generation step and the latter as a *bound spread reduction* step. This issue is briefly discussed and a heuristic to guide the iteration is proposed.

Results in [LM90, MSG89] impose certain restrictions on the allowed set of states at each step of the state generation. In this dissertation, we relax this assumption and show that the general bounding methodology can still be applied.

In Section 3.2, we introduce the model and notation. A brief description of the 'one-step' algorithm is presented in Section 3.3. In Section 3.4, the 'multi-step' bounding algorithm is presented. The bound spread reduction algorithm is presented in Section 3.5. Section 3.6 provides an algorithm to decide whether to generate more of the transition rate matrix and apply the multi-step bounding algorithm or apply the bound spread reduction algorithm to refine the bounds for previously generated states. Up to this point in the exposition, we have make certain simplifying assumptions about the set of states that compose the submodel at each step in the iteration. Section 3.7 describes an extension which removes these assumptions. An example is discussed in Section 3.8.

3.2 Markov Model and Assumptions

Assume a Markov model of a repairable computer system with state space $\mathcal{S}$. $\mathcal{S}$ can be partitioned into two disjoint sets: $\mathcal{O}$ and $\mathcal{F}$, where $\mathcal{O}$ is the set of states in which the system is 'operational' and $\mathcal{F}$ is the set of states in which the system is 'failed'. Let $\mathcal{R}(i)$ be the reward rate associated with state i, $i \in \mathcal{S}$. Let $\mathcal{R}$ be the expected reward rate of the Markov model. We can express $\mathcal{R}$ as:

$$\mathcal{R} = \sum_{i \in \mathcal{S}} \pi(i)\mathcal{R}(i) \tag{3.1}$$

where $\pi(i)$ is the steady state probability of state i. Steady state availability is a special case of this expected reward rate function $\mathcal{R}$ where:

$$\mathcal{R}(i) = \begin{cases} 1 & \text{if } i \in \mathcal{O} \\ 0 & \text{if } i \in \mathcal{F} \end{cases} \tag{3.2}$$

Let n be the number of components in the system being modeled. We can partition the state space $\mathcal{S}$ as follow:

$$\mathcal{S} = \{\mathcal{F}_0 \cup \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_n\}$$

where $\mathcal{F}_i$ contains exactly the states with i failed components. Figure 3.1 represents the transition rate matrix G of the model. In Figure 3.1, $Q_{i,j}$ denotes the submatrix of transitions from $\mathcal{F}_i$ to $\mathcal{F}_j$.

In this dissertation, we make two assumptions concerning the availability model. The first assumption is that the underlying Markov process is irreducible. The second assumption is that the underlying Markov process has a block Hessenberg structure, i.e., $Q_{i,j} = 0$ for $j < i - 1$. This corresponds to an assumption that the probability of two or more components becoming operational in an interval of length Δt is $o(\Delta t)$. It is important to note that this assumption does not preclude multiple repair facilities or other common features of dependability models. Also note that a model in which a dormant component becomes active due to the repair of a second component does not violate the assumption. We

$$\begin{bmatrix}
Q_{00} & Q_{01} & Q_{02} & & \cdots & & Q_{0n} \\
Q_{10} & Q_{11} & Q_{12} & & \cdots & & Q_{1n} \\
0 & Q_{21} & Q_{22} & & \cdots & & \\
0 & 0 & Q_{32} & Q_{33} & & & . \\
. & & 0 & Q_{43} & Q_{44} & & . \\
. & \cdots & & \ddots & & \ddots & . \\
. & & & & & & \\
0 & \cdots & & & 0 & Q_{n,n-1} & Q_{nn}
\end{bmatrix}$$

Figure 3.1: Transition matrix G.

simply do not consider such a dormant component as failed in the definition of the state partitions.

3.3 One-step Bounding Algorithm

In this section, we briefly describe the *one-step* bounding algorithm as reported in [MSG89]. This one-step algorithm will be used as the initial step in the general procedure presented later in this chapter.

Since computer systems are designed with high availability in mind, it is reasonable to expect that most of the components are operational most of the time.

With this in mind, the algorithm utilizes an exact representation of transition rates between states in $\bigcup_{i=0}^{k} \mathcal{F}_i$ for some k and represents the behavior of other states approximately via aggregation [Cou77]. Using aggregation, each subset of states $\mathcal{F}_j$, $k < j \leq n$ will be represented by a single state in the new model. The cardinality of the new model is[1]:

$$\sum_{i=0}^{k} |\mathcal{F}_i| \ + \ (n-k) \tag{3.3}$$

Thus, the state space of the problem can be drastically reduced.

The problem in solving this new model is that the transition rates out of aggregate states are not known. However as will be explained below, one can still compute bounds on the system availability. First we state a result needed from [CS84].

Theorem 3.1 *Let L be any n x n matrix with $L \geq 0$ such that each row sum is less than or equal to 1.*

Let $\beta(L) = \{B \mid B$ is an n x n irreducible stochastic matrix and $B \geq L$. $\}$

Let L_i, $0 \leq i \leq n-1$ be the stochastic matrix equal to L except in the i^{th} column. L_i is matrix L with elements in the i^{th} column increased as necessary to make the matrix stochastic.

Let z_i = the vector of steady state probabilities corresponding to L_i.

Let $\mathcal{V}_L = \{ \ v \mid v$ is the vector of steady state probabilities for some $B \in \beta(L)$ $\}$.

Let $\mathcal{Z}_L = \{v \mid \exists \beta_i, 0 \leq i \leq n-1$ such that $\sum \beta_i = 1, v = \sum_{i=0}^{n-1} \beta_i z_i \ \}$.

Then $\mathcal{V}_L = \mathcal{Z}_L$.

We apply this theorem by associating the matrix L with the substochastic

[1] $|\mathcal{F}_i|$ represents the cardinality of the set $\mathcal{F}_i$

matrix corresponding to transitions between states in $\mathcal{D} = \{\mathcal{F}_0 \cup \cdots \cup \mathcal{F}_k\}$. In this context, the theorem has the following probabilistic interpretation. Consider the system starts in a state in $\mathcal{D}$ and as it evolves, every time the original system would have made a transition out of $\mathcal{D}$ this is instead made a transition to state i in $\mathcal{D}$. For each choice of state i, there is a corresponding stochastic matrix L_i formed by incrementing elements of the i^{th} column for the substochastic matrix of $\mathcal{D}$. The probability vector z_i is the conditional state probability vector under the assumption that each time the set of states in $\mathcal{D}$ is reentered, it is via state i. In [CS86], the authors also note that one need only consider those states in $\mathcal{D}$ into which there is at least one non-zero transition rate from a state in $\overline{\mathcal{D}}$ (i.e., the complement of $\mathcal{D}$). We refer to these states in $\mathcal{D}$ as "return states". In our case, only states in $\mathcal{F}_k$ are possible return states (This is based on the second assumption of our model.). The true conditional state probability vector for $\mathcal{D}$ is a linear combination of the solutions for each matrix, L_i, for each possible return state i. Let $R_\mathcal{D}$ be the reward vector for states in $\mathcal{D}$, and let $\mathcal{A}_\mathcal{D}$ be the system availability conditioned on the system being in $\mathcal{D}$. Then clearly:

$$min_i\{z_i R_\mathcal{D}\} \ \leq \mathcal{A}_\mathcal{D} \leq \ max_i\{z_i R_\mathcal{D}\} \tag{3.4}$$

To obtain the steady state system availability $\mathcal{S}_A$, we need one further result from [CS84], which can be stated as follow:

Theorem 3.2 *Let G be partitioned as in Figure 3.1. Consider an n x n matrix Q_{ag} such that*

$$Q_{ag}[I,J] \ = \ v_I Q_{I,J} \mathbf{1}^\mathbf{T}$$

where v_I is the conditional probability vector for states in set I. If

$$\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n})$$

is the steady state probability vector for A_{ag}, *then* $\mathbf{X_i}, 1 \leq \mathbf{i} \leq n$ *is the steady state probability of being in some state of* $\mathcal{F}_i$ *in the original matrix* G.

The problem with direct application of this result is the number of models that have to be solved; namely, one for each state in $\mathcal{F}_k$ to which there is a non-zero transition from a state in $\mathcal{F}_{k+1}$. This will be impractical for most availability applications since the number of such states may be in the order of thousands. In [MSG89], the concept of state cloning was introduced to help solve this problem. With state cloning, only one submodel needs to be solved at the expense of looser bounds due to the state duplication. In the following, we introduce the important concepts from [MSG89].

Given the transition rate matrix in Figure 3.1, we can partition the matrix into three sets of states:

$$\begin{aligned}
\mathcal{G}_0 &= \{\mathcal{F}_0\} \\
\mathcal{G}_1 &= \{\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_k\} \\
\mathcal{G}_2 &= \{\mathcal{F}_{k+1} \cup \mathcal{F}_{k+2} \cup \cdots \cup \mathcal{F}_n\}
\end{aligned}$$

The transition rate matrix G in Figure 3.1 can be represented by:

$$G = \begin{bmatrix} G_{00} & G_{01} & G_{02} \\ G_{10} & G_{11} & G_{12} \\ 0 & G_{21} & G_{22} \end{bmatrix}$$

with $G_{i,j}$ containing the rates for transitions between states in $\mathcal{G}_i$ and states in $\mathcal{G}_j$.

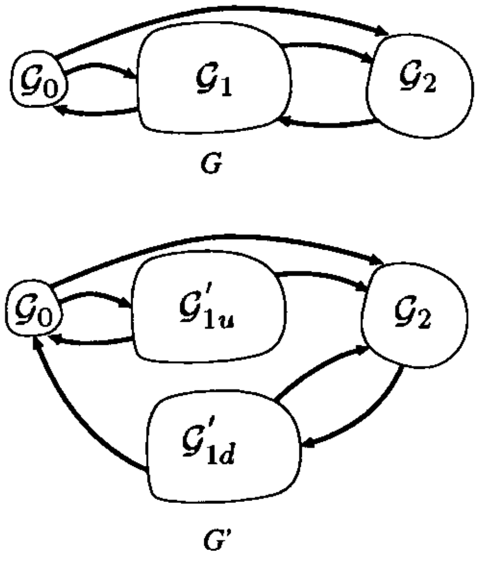

Figure 3.2: Relationship of G and G'.

Figure 3.2 illustrates the concept of state cloning. Applying the technique of state cloning to the states in $\mathcal{G}_1$ of the transition rate matrix G, we have a new transition rate matrix G':

$$G' = \left[\begin{array}{cc|cc} G_{00} & G_{01} & 0 & G_{02} \\ G_{10} & G_{11} & 0 & G_{12} \\ \hline G_{10} & 0 & G_{11} & G_{12} \\ 0 & 0 & G_{21} & G_{22} \end{array}\right]$$

There are two sets of states in G', namely $\mathcal{G}'_{1u}$ and $\mathcal{G}'_{1d}$, that correspond to the set $\mathcal{G}_1$ in G. Each state in $\mathcal{G}_1$ of G maps to a state in $\mathcal{G}'_{1u}$ and to a state in $\mathcal{G}'_{1d}$. The interpretation of cloning can be explained as follows. Assume the system starts in $\mathcal{G}_0$. As components fail and are repaired, the process remains in $\mathcal{G}_0$ and $\mathcal{G}'_{1u}$ until for the first time there are $k+1$ or more failed components. At this point the process is in $\mathcal{G}_2$. When the number of failed components falls to k, the process enters $\mathcal{G}'_{1d}$. The system then evolves in $\mathcal{G}_2$ and $\mathcal{G}'_{1d}$ until the next time it enters $\mathcal{G}_0$. From the construction, it is easy to show that the steady state probability vector of for G is related to the steady state probability vector for G' as follows:

If $[\pi'_0, \pi'_{1_u}, \pi'_{1_d}, \pi'_2]$ is the solution of $\pi' G' = 0$ and $\sum \pi'(i) = 1$,

then $[\pi'_0, \pi'_{1_u} + \pi'_{1_d}, \pi'_2]$ is the solution of $\pi G = 0$ and $\sum \pi(i) = 1$

Let $\mathcal{G}'_{1d} = \bigcup_{i=1}^{k} \mathcal{F}'_i$ where $\mathcal{F}'_i$ is the subset of states of $\mathcal{G}'_{1d}$ in which there are exactly i failed components. Now for each $\mathcal{F}_i$, $i \geq k+1$ form one aggregate state and for each $\mathcal{F}'_i$, $1 \leq i \leq k$ form one aggregate state. Then the transition rate matrix is depicted in Figure 3.3 where the $r_{i,j}$ represent transition rates between aggregate states.

Let us define $\mathcal{D}_1 = \{\mathcal{G}_0 \cup \mathcal{G}'_{1u}\}$ and $\mathcal{C}_1 = \{\mathcal{G}'_{1d}\}$. Note that the transition rate matrix between states in $\mathcal{D}_1$ corresponds to:

$$\begin{bmatrix} G_{00} & G_{01} \\ G_{10} & G_{11} \end{bmatrix}$$

and as illustrated in the rate matrix in Figure 3.3, there is only one state by which $\mathcal{D}_1$ can be entered. Therefore, only one model, L_1, has to be solved to apply Theorem 3.1.

To solve this Markov chain exactly requires determining the transition rates out of the aggregates states. Unfortunately, in general these transition rates can only be found by solving the original model in Figure 3.1. In [MSG89], it is shown that by putting the appropriate bounds on the aggregate transition rates,

$$\left[\begin{array}{cc|cccccc}
G_{00} & G_{01} & 0 & \cdots & 0 & G_{0\;k+1} & \cdots & G_{0n} \\
G_{10} & G_{11} & 0 & \cdots & 0 & G_{1\;k+1} & \cdots & G_{1n} \\
\hline
r_{1'0} & 0 & \bullet & \cdots & r_{1'k'} & r_{1'k+1} & \cdots & r_{1'n} \\
0 & 0 & r_{2'1'} & \bullet & \cdots & r_{2'k+1} & \cdots & r_{2'n} \\
0 & 0 & 0 & r_{3'2'} & \cdots & r_{3'k+1} & \cdots & r_{3'n} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & r_{n\;n-1} & \bullet
\end{array}\right]$$

Figure 3.3: Form of transition matrix after aggregation.

lower bound state probabilities for states in $\mathcal{D}_1$ can be obtained. This result is the subject of the following theorem.

Theorem 3.3 *Consider a Markov process with generator G as depicted in Figure 3.3. By replacing the non-zero aggregate failure rates (i.e., r_{ij} with $j > i$) by upper bounds and the aggregate repair rates (i.e., r_{ij} with $j < i$) with non-zero lower bounds, the Markov chain remains irreducible and the solution of the resulting model yields lower bounds for the stationary state probabilities for the states in $\mathcal{D}_1$.*

Proof : The proof is given in [MSG89] □

Replacing the aggregate transition rates in Figure 3.3 with bounds as described in the above theorem, we obtain the matrix in Figure 3.4 in which the rates indicated by a '+' are replaced by upper bounds (on the actual values) and the rates indicated by '$-$' are replaced by non-zero lower bounds. A trivial upper bound is the sum of all component failure rates and a trivial lower bound is the minimum of all component repair rates. Note that if the original Markov process is irreducible then the constructed Markov process will remain irreducible since no non-zero transition rates are made zero.

$$\left[\begin{array}{cc|cccccc}
G_{00} & G_{01} & 0 & \cdots & 0 & G_{0\ k+1} & \cdots & G_{0n} \\
G_{10} & G_{11} & 0 & \cdots & 0 & G_{1\ k+1} & \cdots & G_{1n} \\
\hline
- & 0 & \bullet & \cdots & + & + & \cdots & + \\
0 & 0 & - & \bullet & \cdots & + & \cdots & + \\
0 & 0 & 0 & - & \cdots & + & \cdots & + \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & - & \bullet
\end{array}\right]$$

Figure 3.4: Form of rate matrix after bounding transition rates.

Once we find lower bounds on the state probabilities for states in $\mathcal{D}_1$, we can express bounds on the steady state system availability $\mathcal{S}_A$ as follows: let $\pi'(i)$ be the stationary state probability for state i with the transition rate matrix in Figure 3.4. Since for all $i \in \mathcal{D}_1$, $\pi'(i)$ is a lower bound on the stationary probability for i in the original model, it follows easily that the bounds are:

$$\sum_{i\in\mathcal{D}_1} \pi'(i)\mathcal{R}(i) \quad \leq \quad S_{\mathcal{A}} \quad \leq \quad \sum_{i\in\mathcal{D}_1} \pi'(i)\mathcal{R}(i) + \left(1 - \sum_{i\in\mathcal{D}_1} \pi'(i)\right) \tag{3.5}$$

$$where \ \ \mathcal{R}(i) \ = \ \begin{cases} 1 & \text{for operational states} \\ \\ 0 & \text{for non-operational states} \end{cases}$$

The idea behind the expression above is that $1 - \sum_{i\in\mathcal{D}_1} \pi'(i)$ is the fraction of time that is not explicitly accounted for states in $\mathcal{D}_1$. The bounds are obtained by assuming that in this fraction of time, the system is either always operational or always failed.

3.4 Multi-step Bounding Algorithm.

In the 'one-step' bounding algorithm described in the previous section, the dimension of the submatrix $\mathcal{D}_1$ is specified a priori and once the steady state availability bounds have been calculated, there is no means provided for further tightening the bounds. In this section, we describe a *multi-step* bounding algorithm [LM90] which partially alleviates this problem. The multi-step algorithm allows incremental generation of more of the transition rate matrix, i.e. at each step, a new portion of the matrix is generated. Further, at each step the results from the previous steps are used to form a transition rate matrix whose solution provides lower bounds on the stationary state probabilities for an additional set of states. This allows us to incrementally improve the bounds on system availability.

Let us introduce the following notation for the multi-step bounding algorithm:

$\mathcal{D}_i$ = set of states which are generated for the i^{th} step of bounding procedure and for which lower bounds on the stationary state probabilities are to be calculated.

$\mathcal{C}_i$ = clone states for the states in $\mathcal{D}_i$.

c_i^j = aggregate state for all the states in $\mathcal{C}_i$ that have exactly j failed components.

$\mathcal{D}'$ = $\{\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_{i-1} \cup \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_{i-1}\}$

d' = aggregate state for all states in $\mathcal{D}'$.

$\mathcal{A}$ = the complement of $\{\mathcal{D}' \cup \mathcal{D}_i\}$. (i.e., the portion of the state space that is unexplored.)

a_i = aggregated state for all the states in $\mathcal{A}$ that have exactly i failed components.

L_i = integer associated with the i^{th} step which denotes the minimum number of failed components for states in $\mathcal{D}_i$.

H_i = integer associated with the i^{th} step which denotes the maximum number of failed components for states in $\mathcal{D}_i$.

$Q_{\mathcal{D}_i,\mathcal{D}_i}$ = transition rate matrix between states in $\mathcal{D}_i$.

$Q_{\mathcal{D}_i,i}$ = transition rate vector from states in $\mathcal{D}_i$ to state i.

$R_{d',\mathcal{D}_i}$ = transition rate vector from aggregate state d' to states in $\mathcal{D}_i$.

$R_{d',\mathcal{C}_i}$ = transition rate vector from aggregate state d' to states in $\mathcal{C}_i$.

$r_{i,j}$ = transition rate from state i to state j.

$\pi_{\mathcal{D}_i/G_i}$ = vector of stationary state probabilities for states in $\mathcal{D}_i$ when the transition rate matrix is G_i.

At any step of the 'multi-step' bounding algorithm, there are three disjoint sets of states. They are $\mathcal{D}', \mathcal{D}_i$ and $\mathcal{A}$. During the i^{th} step of the algorithm, $\mathcal{D}_i$

is composed of all the states with the number of failed components between L_i and H_i. Figure 3.5 illustrates this partitioning of the state space in terms of the transition matrix G.

In the following, we describe a sequence of state space transformations to rate matrix G. We will show that for each state space transformation, state probabilities for states in $\mathcal{D}_i$ are individually bounded from below by each new model in succession. During this sequence of state space transformation, we use the basic aggregation/disaggregation technique described in [Cou77]. One important note is that exact aggregation is not actually required in the computation of the steady state system availability bounds. We merely use the existence of an exact aggregation in the intermediate steps of the development as in the previous section. In the end, we only need bounds on transition rates out of aggregate states.

Figure 3.6 depicts the rate matrix G_1 as the result of the first transformation of G. G_1 corresponds to the cloning of the states in $\mathcal{D}_i$ and this set of cloned states is denoted by $\mathcal{C}_i$. Note that in rate matrix G_1, the submatrix $Q_{\mathcal{C}_i,\mathcal{C}_i}$ is equal to $Q_{\mathcal{D}_i,\mathcal{D}_i}$. A similar transformation was described in the previous section and based on that discussion, it is clear that:

if $[\pi_{\mathcal{D}'/G_1}, \pi_{\mathcal{D}_i/G_1}, \pi_{\mathcal{C}_i/G_1}, \pi_{A/G_1}]$ is the solution of $\pi_1 G_1 = 0$ and $\sum \pi_{G_1}(i) = 1$,

then $[\pi_{\mathcal{D}'/G_1}, \pi_{\mathcal{D}_i/G_1} + \pi_{\mathcal{C}_i/G_1}, \pi_{A/G_1}]$ is the solution of $\pi G = 0$ and $\sum \pi_G(i) = 1$.

Since $\pi_{\mathcal{C}_i/G_1} \geq 0$, the following relationship holds:

$$\pi_{\mathcal{D}_i/G_1} \leq \pi_{\mathcal{D}_i/G} \tag{3.6}$$

Figure 3.7 depicts the next transformation. In this transformation, G_2 is formed from G_1 by applying exact aggregation to the states in $\mathcal{D}'$. Let d' be

$$\begin{bmatrix} Q_{\mathcal{D}'\mathcal{D}'} & Q_{\mathcal{D}'\mathcal{D}_i} & Q_{\mathcal{D}'A} \\ Q_{\mathcal{D}_i\mathcal{D}'} & Q_{\mathcal{D}_i\mathcal{D}_i} & Q_{\mathcal{D}_iA} \\ 0 & Q_{A\mathcal{D}_i} & Q_{AA} \end{bmatrix}$$

Figure 3.5: Rate matrix G. Initial matrix.

$$\begin{bmatrix} Q_{\mathcal{D}'\mathcal{D}'} & Q_{\mathcal{D}'\mathcal{D}_i} & 0 & Q_{\mathcal{D}'A} \\ Q_{\mathcal{D}_i\mathcal{D}'} & Q_{\mathcal{D}_i\mathcal{D}_i} & 0 & Q_{\mathcal{D}_iA} \\ Q_{\mathcal{D}_i\mathcal{D}'} & 0 & Q_{\mathcal{C}_i\mathcal{C}_i} & Q_{\mathcal{D}_iA} \\ 0 & 0 & Q_{A\mathcal{D}_i} & Q_{AA} \end{bmatrix}$$

Figure 3.6: Rate matrix G_1. Introduction of clone states.

the state which represents the aggregation of all states in $\mathcal{D}'$. Because exact aggregation is applied, we have the following relationship:

$$\pi_{\mathcal{D}_i/G_2} = \pi_{\mathcal{D}_i/G_1} \tag{3.7}$$

G_3 in Figure 3.8 is the result of the next state space transformation on G_2. G_3 has a structure similar to that of G_2 except that the transitions from d' to states in $\mathcal{D}_i$ and $\mathcal{C}_i$ are modified. The submatrices $R'_{d'\mathcal{D}_i}$ and $R'_{d'\mathcal{C}_i}$ in G_3 and $R_{d'\mathcal{D}_i}$ in G_2 have the following relationship :

$$R'_{d'\mathcal{D}_i} \;\geq\; \mathbf{0}$$

$$\begin{bmatrix} \bullet & R_{d'\mathcal{D}_i} & 0 & R_{d'A} \\ Q_{\mathcal{D}_i d'} & Q_{\mathcal{D}_i\mathcal{D}_i} & 0 & Q_{\mathcal{D}_i A} \\ Q_{\mathcal{D}_i d'} & 0 & Q_{\mathcal{C}_i\mathcal{C}_i} & Q_{\mathcal{D}_i A} \\ 0 & 0 & Q_{A\mathcal{D}_i} & Q_{AA} \end{bmatrix}$$

Figure 3.7: Rate matrix G_2. After exact aggregation of the states in $\mathcal{D}'$.

$$\begin{aligned} R'_{d'\mathcal{C}_i} &\geq \mathbf{0} \\ R'_{d'\mathcal{D}_i} + R'_{d'\mathcal{C}_i} &= R_{d'\mathcal{D}_i} \end{aligned} \tag{3.8}$$

A probabilistic interpretation is that the original transitions from d' to states in $\mathcal{D}_i$ are each 'split' so that part remains to the corresponding state in $\mathcal{D}_i$ and part goes to the corresponding cloned state in $\mathcal{C}_i$.

From the definition of G_3, we can prove the follow theorem:

Theorem 3.4 $\pi_{\mathcal{D}_i/G_3} \leq \pi_{\mathcal{D}_i/G_2}$

Proof : The proof is given in Appendix A. □ .

In the next transformation, we apply exact aggregation to the subsets of states in $\mathcal{C}_i$ and $\mathcal{A}$. We form one aggregate state for each subset $\mathcal{F}_j$ in $\mathcal{C}_i$ and $\mathcal{F}_j$ in $\mathcal{A}$. The result of this transformation is rate matrix G_4 as depicted in Figure 3.9. Note that we permuted the ordering of the state d' and the set of states $\mathcal{D}_i$ so that the matrix has the same form as the rate matrix in Figure 3.4. Since we

$$\begin{bmatrix} \bullet & R'_{d'\mathcal{D}_i} & R'_{d'\mathcal{C}_i} & R_{d'A} \\ Q_{\mathcal{D}_i d'} & Q_{\mathcal{D}_i\mathcal{D}_i} & 0 & Q_{\mathcal{D}_i A} \\ Q_{\mathcal{D}_i d'} & 0 & Q_{\mathcal{C}_i\mathcal{C}_i} & Q_{\mathcal{D}_i A} \\ 0 & 0 & Q_{A\mathcal{D}_i} & Q_{AA} \end{bmatrix}$$

Figure 3.8: Rate matrix G_3. Modified rates from state d'.

applied exact aggregation in forming G_4, the following relationship holds:

$$\pi_{\mathcal{D}_i/G_4} = \pi_{\mathcal{D}_i/G_3} \tag{3.9}$$

Since the rate matrix in Figure 3.4 and the rate matrix G_4 in Figure 3.9 have the same form, Theorem 3.3 can be apply. Specifically, if the elements shown in Figure 3.10 as '$+$' are replaced by upper bounds on those rates and the elements shown as '$-$' are replaced by non-zero lower bounds. The Markov process remains irreducible and the solution for the stationary state probabilities will yield a lower bound for the state probabilities for states in $\mathcal{D}_i$. Therefore, the following relationship holds:

$$\pi_{\mathcal{D}_i/G_5} \leq \pi_{\mathcal{D}_i/G_4} \tag{3.10}$$

From this sequence of transformation, we can conclude that:

$$\pi_{\mathcal{D}_i/G_5} \leq \pi_{\mathcal{D}_i/G} \tag{3.11}$$

In terms of state space cardinality, clearly G_5 has a much reduced state space compared with that of G. The remaining question is how to provide bounds for

$$\left[\begin{array}{cc|cccc|cccc}
Q_{\mathcal{D}_i,\mathcal{D}_i} & Q_{\mathcal{D}_i,d'} & 0 & 0 & \cdots & 0 & Q_{\mathcal{D}_i,a_{H_i+1}} & Q_{\mathcal{D}_i,a_{H_i+2}} & \cdots & Q_{\mathcal{D}_i,a_n} \\
R'_{d',\mathcal{D}_i} & \bullet & r_{d',c_i^{L_i}} & r_{d',c_i^{L_i+1}} & \cdots & r_{d',c_i^{H_i}} & r_{d',a_{H_i+1}} & r_{d',a_{H_i+2}} & \cdots & r_{d',a_n} \\
\hline
0\ldots0 & r_{c_i^{L_i},d'} & \bullet & r_{c_i^{L_i},c_i^{L_i+1}} & \cdots & r_{c_i^{L_i},c_i^{H_i}} & r_{c_i^{L_i},a_{H_i+1}} & r_{c_i^{L_i},a_{H_i+2}} & \cdots & r_{c_i^{L_i},a_n} \\
0\ldots0 & 0 & r_{c_i^{L_i+1},c_i^{L_i}} & \bullet & \cdots & r_{c_i^{L_i+1},c_i^{H_i}} & r_{c_i^{L_i+1},a_{H_i+1}} & r_{c_i^{L_i+1},a_{H_i+2}} & \cdots & r_{c_i^{L_i+1},a_n} \\
0\ldots0 & 0 & 0 & r_{c_i^{L_i+2},c_i^{L_i+1}} & \bullet & r_{c_i^{L_i+2},c_i^{H_i}} & r_{c_i^{L_i+2},a_{H_i+1}} & r_{c_i^{L_i+2},a_{H_i+2}} & \cdots & r_{c_i^{L_i+2},a_n} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0\ldots0 & \vdots & 0 & \cdots & \cdots & \bullet & r_{c_i^{H_i},a_{H_i+1}} & \cdots & \cdots & r_{c_i^{H_i},a_n} \\
\hline
0\ldots0 & 0 & 0 & 0 & 0 & r_{a_{H_i+1},c_i^{H_i}} & \bullet & r_{a_{H_i+1},a_{H_i+2}} & \cdots & r_{a_{H_i+1},a_n} \\
\vdots & 0 & 0 & 0 & 0 & 0 & r_{a_{H_i+2},a_{H_i+1}} & \bullet & \cdots & r_{a_{H_i+2},a_n} \\
\vdots & 0 & 0 & 0 & 0 & 0 & 0 & r_{a_{H_i+3},a_{H_i+2}} & \bullet & r_{a_{H_i+3},a_n} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
0\ldots0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \bullet
\end{array}\right]$$

Figure 3.9: Rate matrix G_4. Exact aggregation of states in $\mathcal{C}_i$ and $\mathcal{A}_i$.

$$\left[\begin{array}{cc|ccccc|cccc}
Q_{\mathcal{D}_i,\mathcal{D}_i} & Q_{\mathcal{D}_i,d'} & 0 & 0 & \cdots & \cdots & 0 & Q_{\mathcal{D}_i,A_{H_i+1}} & Q_{\mathcal{D}_i,A_{H_i+2}} & \cdots\cdots & Q_{\mathcal{D}_i,A_n} \\
R'_{d',\mathcal{D}_i} & \bullet & + & + & \cdots & \cdots & + & + & + & \cdots\cdots & + \\
\hline
0\ldots0 & - & \bullet & + & \cdots & \cdots & + & + & + & \cdots\cdots & + \\
0\ldots0 & 0 & - & \bullet & \cdots & \cdots & + & + & + & \cdots\cdots & + \\
\vdots & \vdots & 0 & - & \bullet & \cdots & + & + & \cdots & \cdots\cdots & + \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\ \vdots & \vdots \\
0\ldots0 & \vdots & 0 & \cdots\cdots & 0 & - & \bullet & + & \cdots & \cdots\cdots & + \\
\hline
0\ldots0 & 0 & 0 & 0 & 0 & \cdots & - & \bullet & + & \cdots\cdots & + \\
\vdots & 0 & 0 & 0 & 0 & \cdots & 0 & - & \bullet & \cdots\cdots & + \\
\vdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & - & \bullet\ \cdots & + \\
\vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots\ \ddots & \vdots \\
0\ldots0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots\ - & \bullet
\end{array}\right]$$

Figure 3.10: Rate matrix G_5. Replacement of transition rates with bounds.

rates from state d' to $\mathcal{D}_i$ and $\mathcal{C}_j^i$ for $L_i \leq j \leq H_i$. To answer the question, let us define the following:

$\pi_{\mathcal{D}'}$ = exact steady state probability vector for states in $\mathcal{D}'$.

$\pi'_{\mathcal{D}'}$ = lower bound steady state probability vector for states in $\mathcal{D}'$ (which we have computed in previous steps).[2]

a = sum of the lower bound state probabilities for all those states we have generated so far.

$\mathbf{1}$ = column vector of 1's.

From Equation (3.8) and the rate matrix G_5, it is clear that we obtain lower bound steady state probabilities for states in $\mathcal{D}_i$ if the following relationships hold:

$$\begin{aligned} R_{d'\mathcal{D}_i} &\geq R'_{d'\mathcal{D}_i} \\ R_{d'\mathcal{D}_i} &\leq R'_{d'\mathcal{D}_i} + R'_{d'\mathcal{C}_i} \end{aligned} \tag{3.12}$$

Since $\pi'_{\mathcal{D}'}$ is the lower bound steady state probability vector, it follows immediately that $R'_{d',\mathcal{D}_i}$ can be expressed as:

$$\begin{aligned} R_{d',\mathcal{D}_i} &= [\pi_{\mathcal{D}'}\mathbf{1}]^{-1}\pi_{\mathcal{D}'}Q_{\mathcal{D}',\mathcal{D}_i} \\ &\geq [\pi'_{\mathcal{D}'}\mathbf{1} + (1-a)]^{-1}\pi'_{\mathcal{D}'}Q_{\mathcal{D}',\mathcal{D}_i} = R'_{d',\mathcal{D}_i} \end{aligned} \tag{3.13}$$

which provides a lower bound for transition rates from state d' to the states in $\mathcal{D}_i$.

For the rates r_{d',c_i^j}, $L_i \leq j \leq H_i$, let us define the following:

[2] If the lower bound steady state probability of some $\mathcal{C}_k$, $1 \leq k \leq i-1$, are unknown, 0 will be used as lower bound. In the next Section, we will describe how we obtain lower bound steady state probabilities for states in $\mathcal{C}_k$.

$\mathcal{D}_i^j$ = states in $\mathcal{D}_i$ with exactly j failed components.

$R'_{d',\mathcal{D}_i^j}$ = transition rate vector from state d' to states in $\mathcal{D}_i^j$.

$Q_{\mathcal{D}',\mathcal{D}_i^j}$ = transition rate matrix from states $\mathcal{D}'$ to states in $\mathcal{D}_i^j$.

r_{max} = maximum entry in vector $Q_{\mathcal{D}',\mathcal{D}_j^i}\mathbf{1}$.

The r_{d',c_i^j}, rate from aggregate state d' to aggregate c_i^j is easily seen to satisfy the relationship:

$$\begin{aligned} r_{d',c_i^j} &= MIN\,\{sum\ \ of\ \ all\ \ failure\ \ rates, \\ &\quad [\pi'_{\mathcal{D}'}\mathbf{1}]^{-1}\left[\pi'_{\mathcal{D}'}Q_{\mathcal{D}',\mathcal{D}_i^j}\mathbf{1} + (1-a)r_{max}\right] - R'_{d',\mathcal{D}_i^j}\mathbf{1}\Big\} \end{aligned} \tag{3.14}$$

which provides a upper bound for transition rates from state d' to each aggregate states c_i^j, $L_i \le j \le H_i$.

3.5 Bound Spread Reduction.

In the previous two sections, we have described a *one-step* and a *multi-step* procedure for bounding the steady state availability of a repairable computer system. In these bounding procedures, *errors*[3] occur at each step. These errors can be classified according to their sources:

1. by considering (a) the clone states, $\mathcal{C}_i$ and (b) the aggregate states, A to have reward of 0 or 1 in the evaluation of the availability bounds.

2. the difference between the lower bounds and the actual values of the stationary state probabilities of the 'detailed states' in $\mathcal{D}_i$.

The contribution to the bound spread by components, (1a) and (2) are not reduced by successive application of the multi-step algorithm. In this section we show how we can *reduce* these errors and obtain tighter bounds by reevaluating previously calculated bounds for state probabilities. The bound spread reduction algorithm we propose is iterative in nature. We will first reduce the error associated with clone states by obtaining lower bounds on their stationary state probabilities. Once we obtain these lower bound state probabilities, an improved estimate of the transition rates out of the aggregate clone states can be computed. With these improved transition rates, we can reduce the error in the stationary state probabilities of the 'detailed states' $\mathcal{D}_i$. An important point is that the bound spread reduction algorithm does not require generating more of the transition matrix. (It does require reusing the previously generated portions of the transition rate matrix.) In section 3.5.1, we present the approach to reducing the

[3] error is defined to be the spread of the system availability bounds.

error associated with clone states and in section 3.5.2, we present the approach to obtaining improved lower bound state probabilities for the 'detailed states', $\mathcal{D}_i$.

3.5.1 Bound spread reduction for the clone states.

Until now we have assigned a reward of 0 or 1 to all aggregate clone states in the computation of the system availability bounds. In this way, we never had to be concerned with the state probabilities for individual clone states in $\mathcal{C}_i$. To reduce this source of error, we would like to obtain a lower bound for the state probability of each individual clone state. To do this, we will make use of the fact that the clone states have exactly the same transition structure as the detailed states, $\mathcal{D}_i$. Since this portion of transition rate matrix was generated in the previous step, we can use it to compute a lower bound on the state probabilities of the individual clone states.

Assume that at $i+1^{th}$ step of the multi-step procedure, we have already obtained lower bounds on the state probabilities for all detail states in $\mathcal{D}_{i+1}$ and we now want to find the lower bound steady state probabilities for all clone states in $\mathcal{C}_i$. Let us define the following notation:

$\pi_{\mathcal{C}_i}$ = steady state probability vector for the clone states in $\mathcal{C}_i$.

$\pi_{\mathcal{D}_{i+1}^{L_{i+1}}}$ = state probability vector for the detail states in $\mathcal{D}_{i+1}$ with exactly L_{i+1} failed components.

$Q_{\mathcal{D}_{i+1}^{L_{i+1}},\mathcal{C}_i}$ = transition rate matrix corresponding to transitions from states in $\mathcal{D}_{i+1}^{L_{i+1}}$ to states in $\mathcal{C}_i$.

From the flow conservation equation, we have:

$$\pi_{\mathcal{C}_i} Q_{\mathcal{D}_i,\mathcal{D}_i} + \pi_{\mathcal{D}_{i+1}^{L_{i+1}}} Q_{\mathcal{D}_{i+1}^{L_{i+1}},\mathcal{C}_i} = \mathbf{0} \tag{3.15}$$

Note that in the previous step we obtained lower bounds on the state probabilities $\pi_{\mathcal{D}_{i+1}^{L_{i+1}}}$ and have generated the transition rate structure corresponding to transitions between states that have from L_i to H_i failures. We now show that by applying an iterative solution method (e.g., Jacobi or Gauss-Seidel Iterative method [Var62]), we obtain a lower bound on the state probabilities of the clone states. In the remainder of this section, we formulate the iterative procedure using the Gauss-Seidel iterative method. Note that $\pi_{\mathcal{D}_{i+1}^{L_{i+1}}}$ is constant in this algorithm (the lower bound state probabilities obtained from the previous bounding step). We will show that the iterative procedure:

1. converges and converges to a *unique* solution.

2. converges from below.

3. converges monotonically.

4. the solution (fixed point) is a lower bound of the exact state probabilities of the clone states.

These characteristics are especially interesting because they indicate that the iterative process can be *terminated at any step* and the current values are guaranteed to be lower bounds on the state probabilities of the clone states.

Let us rewrite Equation (3.15) as:

$$-Q^T_{\mathcal{D}_i,\mathcal{D}_i} \pi^T_{\mathcal{C}_i} = Q^T_{\mathcal{D}_{i+1}^{L_{i+1}},\mathcal{C}_i} \pi^T_{\mathcal{D}_{i+1}^{L_{i+1}}} \tag{3.16}$$

which has the form of a linear system $\mathbf{A}\ \mathbf{x} = \mathbf{b}$ (with $\mathbf{A} = -Q^T_{\mathcal{D}_i,\mathcal{D}_i}$ and $\mathbf{x} = \pi^T_{\mathcal{C}_i}$). Note that each diagonal element of $\mathbf{A}$ is the absolute value of transition rate out of the associated clone state and the off diagonal elements are the 'negated' transition rates from one clone state to another clone state. Let $\mathbf{A} = [\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A} - \mathbf{U}_\mathrm{A}]$ where $\mathbf{D}_\mathrm{A}$ is a diagonal matrix and $\mathbf{L}_\mathrm{A}$ and $\mathbf{U}_\mathrm{A}$ are lower and upper triangular matrices respectively. The Gauss-Seidel iteration can be written as[4]:

$$\begin{aligned} \mathbf{x}^{(k)} &= [(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}]\mathbf{x}^{(k-1)} + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b} \\ with\ \mathbf{x}^{(0)} &= \mathbf{0} \end{aligned} \tag{3.17}$$

A necessary and sufficient condition for the above iterative process to converge to an unique solution is for the spectral radius $\rho[(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}]$ to be less than 1 [BF88]. Since $\mathbf{A}$ and $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})$ are non-singular M-matrices, their inverses are non-negative matrices, and $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})$ and $\mathbf{U}_\mathrm{A}$ form a regular splitting of matrix $\mathbf{A}$ [Var62]. Therefore the spectral radius $\rho[(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}]$, is less than 1. Therefore the iterative process does converge to a unique solution. In the following, we will show that the iterative procedure also has the other claimed characteristics.

Lemma 3.1 *The proposed iterative procedure converges from below.*

Proof : Since the iterative process converges to a unique solution $\mathbf{x}^*$. From Equation (3.17), we have:

$$\mathbf{x}^* = \left[(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}\right]\ \mathbf{x}^* + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b}$$

[4] In order to guarantee the claimed characteristics, a initial vector of zero is chosen. There exist other initial vectors that can speed up the convergence rate but we have not been able to show the claimed characteristics can be guaranteed with these other choices.

Let $\mathbf{e}^{(k)}$ be the error vector at the k^{th} iteration. Then:

$$\begin{aligned}\mathbf{e}^{(k)} &= \mathbf{x}^* - \mathbf{x}^{(k)}\\ &= (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}(\mathbf{x}^* - \mathbf{x}^{(k-1)})\\ &= (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}(\mathbf{e}^{(k-1)})\end{aligned}$$

In order to have convergence from below, we need $\mathbf{e}^{(k)} \geq \mathbf{0}$ for all k. Since $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}$ and $\mathbf{U}_\mathrm{A}$ are nonnegative matrices and the initial estimate is a lower bound vector, it follows easily that $\mathbf{e}^{(k)} \geq 0$ for all k. □

Lemma 3.2 *The proposed iterative procedure converges monotonically.*

Proof: To show that we have an improved bound at each iteration, it is sufficient to show that $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \geq \mathbf{0}$ for all k. This can be easily proved by induction.

Basis : For $k = 0$. From Equation (3.17), we see that $\mathbf{x}^{(0)} = \mathbf{0}$ Also, we see that $\mathbf{x}^{(1)} = (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b} \ \geq \ \mathbf{0}$. Therefore $\mathbf{x}^{(1)} - \mathbf{x}^{(0)} \ \geq \ \mathbf{0}$.

Induction : Assume $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \ \geq \ \mathbf{0}$ for $k \leq n$. For $k = n+1$, we have:

$$\mathbf{x}^{(n+2)} - \mathbf{x}^{(n+1)} = (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}(\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) \ \geq \ \mathbf{0}$$

The inequality holds since $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}$ and $\mathbf{U}_\mathrm{A}$ are nonnegative matrices. □

Lemma 3.3 *The fixed point of the proposed iterative procedure is a lower bound on the exact state probabilities of the clone states.*

Proof : Let $\mathbf{x}'$ be the exact state probabilities vector for the clone states. We have to show that $\mathbf{x}' - \mathbf{x}^* \geq \mathbf{0}$. Let $\mathbf{b}'$ contain the *exact* rates into the clone states from states in $\mathcal{D}_{i+1}^{L_i+1}$. Then:

$$\mathbf{A}\ \mathbf{x}^* = \mathbf{b}$$

$$\begin{aligned} \mathbf{A}\,\mathbf{x}' &= \mathbf{b}' \\ \mathbf{b}' - \mathbf{b} &\geq \mathbf{0} \end{aligned}$$

The above inequality holds because we computed **b** from lower bound state probabilities for states in $\mathcal{D}_i^{L_i+1}$. It is easily seen that:

$$\begin{aligned} \mathbf{x}' - \mathbf{x}^* &= \mathbf{A}^{-1}\mathbf{b}' \;-\; \mathbf{A}^{-1}\mathbf{b} \\ &= \mathbf{A}^{-1}(\mathbf{b}' - \mathbf{b}) \qquad \geq \mathbf{0} \end{aligned}$$

Since $(\mathbf{b}' - \mathbf{b}) \geq \mathbf{0}$ and $\mathbf{A}^{-1} \geq \mathbf{0}$. □

To summarize, the algorithm for bound spread reduction for the clone states in $\mathcal{C}_i$ is as follows:

```
procedure   Bound Spread Reduction for $\mathcal{C}_i$
begin
    Let $\pi_{\mathcal{C}_i} = \mathbf{0}$;
    do
        apply the iterative procedure embodied in
                Equation (3.17);
    while (specific tolerance is satisfied) ;
end
```

3.5.2 Bound spread reduction for the detail states.

Recall that in computing lower bounds for the state probabilities of the detail states $\mathcal{D}_i$, we used upper bound failure rates (e.g. sum of the failure rates) and lower bound repair rates (e.g: minimum repair rate) for the aggregates. Using the procedure described in Section 3.5.1, we can obtain better lower bound estimates for the stationary state probabilities of the clone states. This in turn can be used to generate tighter bounds on the aggregate transition rates out of the clone states and thereby obtain improved lower bounds for the state probabilities of the detailed states, $\mathcal{D}_i$.

Let us define the following notation:

$r(c_i^j, c_i^k)$ = transition rate from the clone aggregate c_i^j to clone aggregate c_i^k.

a = sum of the lower bound state probabilities for all those states we have generated so far.

$\pi_{\mathcal{C}_i^k}$ = lower bound state probability vector for clone states in $\mathcal{C}_i$ with exactly k failed components.

r_{max} = maximum entry in vector $Q_{\mathcal{D}_i^k, \mathcal{D}_i^m}\mathbf{1}$.

Since we already computed lower bounds for the state probability of each clone state, *improved* lower bounds on the 'repair rates' between the aggregates can be obtained as follows:

$$\begin{aligned} r(c_i^k, c_i^{k-1})^- \;&=\; MAX\,\{minimum\ repair\ rate, \\ &\qquad [\pi_{\mathcal{C}_i^k}\mathbf{1} + (1-a)]^{-1}\pi_{\mathcal{C}_i^k}Q_{\mathcal{D}_i^k, \mathcal{D}_i^{k-1}}\mathbf{1}\big\} \end{aligned} \tag{3.18}$$

while an *improved* upper bound on the failure rate from aggregate c_i^k to aggregate c_i^m is as follows:

$$\begin{aligned} r(c_i^k, c_i^m)^+ \;&=\; MIN\,\{sum\ of\ all\ failure\ rates, \\ &\qquad (\pi_{\mathcal{C}_i^k}\mathbf{1})^{-1}[\pi_{\mathcal{C}_i^k}Q_{\mathcal{D}_i^k, \mathcal{D}_i^m}\mathbf{1} + (1-a)r_{max}]\big\} \end{aligned} \tag{3.19}$$

Equations (3.18) and (3.19) follow easily from considering the conditional transition rates between aggregates based on upper and lower bounds conditional state probabilities. For example, in equation (3.18):

$$[\pi_{\mathcal{C}_i^k}\mathbf{1} + (1-a)]^{-1}\pi_{\mathcal{C}_i^k}$$

is a lower bound on the conditional state probability vector for states in $\mathcal{C}_i^k$.

To summarize, the bound spread reduction algorithm for states in $\mathcal{D}_i$ is as follow:

```
procedure   Bound Spread Reduction for $\mathcal{D}_i$
begin
        Based on the rate matrix $G_5$ in Section 3.4,
            for each pair $(c_i^j, c_i^{j-1})$ where $c_i^j, c_i^{j-1} \in \mathcal{C}_i$, compute the
                improved clone aggregate repair rate
                $r(c_i^j, c_i^{j-1})$ by Equation (3.18);
            for each pair $(c_i^j, c_i^k)$ where $c_i^j, c_i^k \in \mathcal{C}_i$, compute the
                improved clone aggregate failure rate
                $r(c_i^j, c_i^k)$ by Equation (3.19);
            using the previous fixed point of $\pi_{\mathcal{D}_i}$ computed from the
                multi-step bounding algorithm as a initial vector,
                compute the improved lower bound state
                probability vector of $\pi_{\mathcal{D}_i}$;
end
```

3.5.3 Bound spread reduction algorithm.

The above bound spread reduction procedure will give us improved lower bounds on the state probabilities for detailed states in $\mathcal{D}_i$ and clone states in $\mathcal{C}_i$ corresponding to *step i* of the bounding process. We can repeat the reduction procedure for detail states and clone states corresponding to steps $i-1, i-2, \ldots, 1$ of the bounding process. Since we obtain a better bound for all states in bounding step 1, we can go forward again and apply the multi-step bounding procedure to obtain better state probability bounds for states corresponding to bounding steps $2, 3, \ldots, i+1$. The complete bound spread reduction strategy can therefore be stated as follows:

```
procedure    Bound Spread Reduction Algorithm
begin
        for $j = i$ to 1 do
        begin
                Apply the algorithm in Section 3.5.1 to reduce the
                        bound spread contribution for clone states in $\mathcal{C}_j$;
                Apply the algorithm in Section 3.5.2 to reduce the
                        bound spread contribution for clone states in $\mathcal{D}_j$;
        end
        for $j = j + 1$ to $i + 1$ do
        begin
                Apply the multi-step bounding algorithm described in
                        Section 3.4 to obtain an improved lower
                        bound in $\mathcal{D}_j$;
        end
end
```

Clearly this reduction strategy can be applied repeatedly to obtain better availability bounds. Also, each time we apply the iterative procedure to improve the lower bounds on the stationary state probabilities for the clone states, the initial vector for the clone state probabilities can be the estimated fixed point solution vector from the previous iterative procedure. Since this estimated fixed point solution vector is a lower bound to the exact solution vector, by using it as a starting vector, we not only reduce the number of iterations but also preserve the characteristics we proved in Lemma 5.1 to Lemma 5.3. The preservation of the characteristics in Lemma 5.1 and 5.3 is trivial. In the following lemma, we show that by using the estimated fixed point solution vector from the previous iterative procedure, we preserve the monotonic convergence characteristic.

Let us define the following:

$\mathbf{b}$ = the vector of conditional rates from $\mathcal{D}_{i+1}$ to $\mathcal{C}_i$ computed using the lower bound state probabilities $\pi_{\mathcal{D}_{i+1}^{L_{i+1}}}$.

$\pi'_{\mathcal{D}_{i+1}^{L_{i+1}}}$ = the vector of new lower bound state probabilities computed in the last step of the reduction strategy, where $\pi'_{\mathcal{D}_{i+1}^{L_{i+1}}} \geq \pi_{\mathcal{D}_{i+1}^{L_{i+1}}}$.

$\mathbf{b}_2$ = the vector of conditional rates from $\mathcal{D}_{i+1}$ to $\mathcal{C}_i$ computed using $\pi'_{\mathcal{D}_{i+1}^{L_{i+1}}}$.

Lemma 3.4 *Using the estimated fixed point solution vector from the previous iterative procedure, the monotonic convergence characteristic is preserved.*

Proof : In the previous step of the reduction strategy we obtained improved lower bounds for the detailed states in bounding step $i+1$. This implies that we have improved lower bounds for the stationary state probabilities of states with L_{i+1} failed components, $\pi'_{\mathcal{D}_{i+1}^{L_{i+1}}}$. Based on these improved lower bounds, we can obtain a new vector $\mathbf{b}_2$ which can be expressed as:

$$\mathbf{b}_2 \;=\; \mathbf{b} + \mathbf{b}_2' \qquad \textit{where} \quad \mathbf{b}_2' \geq \mathbf{0}$$

Similar to Lemma 5.5, we have to show $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \geq \mathbf{0}$ for all k. Again, we prove this by induction. Let $\mathbf{x}^{(0)} = \tilde{\mathbf{x}}$ where $\tilde{\mathbf{x}}$ is the estimated fixed point solution from the previous iterative procedure. **Basis :** For $k = 0$. From (3.17), we have:

$$\begin{aligned}
\mathbf{x}^{(1)} &= \left[(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}\right]\ \mathbf{x}^{(0)} + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b}_2 \\
&= \left[(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}\right]\ \tilde{\mathbf{x}} + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b} + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b}_2' \\
&= \tilde{\mathbf{x}} + \mathbf{x}' + (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{b}_2'
\end{aligned}$$

Since $\mathbf{x}' \geq \mathbf{0}$, $\mathbf{b}_2' \geq \mathbf{0}$ and $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1} \geq \mathbf{0}$, therefore $\mathbf{x}^{(1)} - \tilde{\mathbf{x}} \geq \mathbf{0}$. **Induction :** Assume $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \geq \mathbf{0}$ for $k \leq n$. For $k = n + 1$, we have:

$$\mathbf{x}^{(n+2)} - \mathbf{x}^{(n+1)} = (\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}\mathbf{U}_\mathrm{A}(\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) \geq \mathbf{0}$$

The inequality holds because $(\mathbf{D}_\mathrm{A} - \mathbf{L}_\mathrm{A})^{-1}$ and $\mathbf{U}_\mathrm{A}$ are nonnegative matrices.

□

Lastly, let us estimate the cost of the bound spread reduction procedure after the k^{th} step of the bounding process. Let $|\mathcal{D}_j|$ be the number of detailed states during the j^{th} step bounding algorithm, then the cost of one iteration of the bound spread reduction procedure is:

$$o(\sum_{j=1}^{k} k_j|\mathcal{D}_j|)$$

where k_j is a constant multiplier which is a function of the number of non-zero entries in the matrix $\mathcal{D}_j$ and the number of iterations for the algorithm to converge. In [SG85b], it is reported that the number of iterations often ranges between 20 and 100. Since the starting probability vector is the estimated fixed point solution vector computed from the previous step, the number of iterations to compute the new probability vector is also drastically reduced in most cases for successive iterations.

Although this reduction strategy can be applied repeatedly, there are diminishing returns in successive iterations. It is of interest then to estimate when it is better to repeat the bound spread reduction algorithm and when it is better to generate more of the transition rate matrix corresponding to unexplored states. This issue is discussed in the next section.

3.6 Decision criteria for backward iteration or forward generation

Although tighter availability bounds can be obtained either going forward (i.e., by generating more of the transition rate matrix) and applying the multi-step bounding algorithm or by going backward (i.e., reducing the errors accumulated in the previous steps) and applying the bound spread reduction algorithm, the computational cost and potential gain for these two choices are quite different. For the multi-step bounding algorithm, we have to consider the following:

- computational cost for state generation.
- storage cost for the newly generated transition matrix.
- computational cost for evaluating steady state probabilities in the detail states.

For the bound spread reduction algorithm, the transition matrices were generated in the previous steps, therefore the only cost is the computation cost of the bound spread reduction process and the cost of retrieving the transition matrix from secondary storage if they do not all fit in main memory concurrently.

In order to decide which algorithm to apply, we have to also compare their respective *potential gains*. We define the potential gain as the fractional improvement in the spread between the upper and lower availability bounds. For the bound spread reduction algorithm, the potential gain comes from improved bounds on the detailed states and clone states. In the forward direction, the potential gain comes from the ability to obtain lower bounds for additional states. Although we can apply the bound spread reduction algorithm repeatedly to reduce the errors, the potential gain for each successive application exhibits diminishing returns. On the other hand, going forward will require generating more of the transition rate matrix. But since the distribution of state probabilities is skewed these newly generated states may not make a significant contribution to bound reduction.

Based on the above discussion, we see that the problem of making an optimal decision is not trivial. One important requirement for the decision algorithm is that the computation cost should be much less costly than the multi-step bounding algorithm or the bound spread reduction algorithms themselves. Since we always obtain improved bounds regardless of the decision, the worst possible effect is some inefficiency. Although finding an optimal decision algorithm is an interesting theoretical issue, we conjecture that truly optimal decision algorithm will be very costly to implement and finding the optimal decision algorithm is beyond the scope of this dissertation. Nevertheless, a decision algorithm is required to implement the complete bounding methodology. The heuristic decision algorithm we present is not meant to be optimal in any sense. It is a simple, common sense heuristic which we have found to work well in practice. In the following, we described this heuristic decision criteria.

Let us define the following notation:

$s_f(i)$ = difference between the upper and lower bound availability after the i^{th} application of the forward step algorithm.

$s_b(i)$ = difference between the upper and lower bound availability after the i^{th} application of the bound spread reduction algorithm.

s_c = current difference between the upper and lower bound availability.

$g_f^{'}$ = estimated gain if the multi-step bounding algorithm is applied.

$g_b^{'}$ = estimated gain if the bound spread reduction algorithm is applied.

In deciding whether to apply the forward step algorithm for the $(i+1)^{th}$ time, we estimate the gain $g_f^{'}$ as follows:

$$g_f^{'} = \frac{[s_f(i-1) - s_f(i)]}{s_f(i-1)} s_c \tag{3.20}$$

with the first term representing the fractional potential gain from the previous application of the forward step algorithm. In essence, $g_f^{'}$ estimates that, if the multi-step algorithm is applied, the gain will be the same fraction of the bound spread as was achieved by the previous step.

The estimated gain for going backward after application of the j^{th} bound spread reduction bounding algorithm, $g_b^{'}$ is as follows:

$$g_b^{'} = \frac{[s_b(j-1) - s_b(j)]}{s_b(j-1)} s_c \qquad for\ j \geq 1 \tag{3.21}$$

with $s_b(0) = s_c$. The first term represents the fractional potential gain for the previous iteration of the bound spread reduction algorithm and $g_b^{'}$ estimates that the potential gain will have the same reduction ratio compared with the previous application.

In the following, we propose a simple heuristic algorithm for deciding the next step in the procedure. The algorithm is applied after each forward step. Note that one (backward) bound spread reduction is always applied before a decision is made.

```
procedure    decision algorithm
begin
        Apply the bound spread reduction algorithm described in
                Section 3.5.3 for one iteration;
        Compute $g_f'$ and $g_b'$;
        while ($g_f' \leq g_b'$)
        begin
                Apply bound spread reduction algorithm described from
                                Section 3.5.3 once;
                Compute $g_f'$ and $g_b'$;
        end
        Apply multi-step algorithm from Section 3.4;
end
```

In essence, the decision algorithm is biased toward reducing the accumulated errors from the previous steps. By doing this, it also avoids the state generation cost and the storage cost of the multi-step bounding algorithm. The decision algorithm cost is clearly trivial. A detailed example in Section 3.8 illustrates the application and effectiveness of the heuristic.

The global algorithm is as follows:

```
procedure    Global Bounding Algorithm
```

```
begin
    $i = 1$;
    Based the one-step algorithm described in Section 3.3, generate
        lower bounds on the stationary state probabilities for
        states in $\mathcal{F}_0$ through $\mathcal{F}_{H_1}$ and compute the system
        availability bounds;
    if  (system availability bound is tight enough)
        stop;
    while ( system availability bound not tight enough) do
    begin
        $i = i + 1$;
        Based on the multi-step algorithm described in Section 3.4,
            generated the portion of the rate matrix
            corresponding to $\mathcal{F}_{L_i}$ and $\mathcal{F}_{H_i}$ (with $L_i = H_{i-1} + 1$)
            and compute the system availability bounds;
        if (system availability bound is tight enough)
          stop;
        Apply Bound Spread Reduction algorithm described in
            Section 3.5.3 and compute the system availability
            bounds;
        if (system availability bound is tight enough)
          stop;
        compute $g_f^{'}$ and $g_b^{'}$;
        while ($g_f^{'} \leq g_b^{'}$)
        begin
```

```
                Apply Bound Spread Reduction algorithm
                        described in 3.5.3 and compute the system
                        availability bounds;
                if (system availability bound is tight enough)
                   stop;
                compute $g_f'$ and $g_b'$;
            end
        end
    end
```

Thus, this global bounding algorithm provides a method for piecewise generation of the transition matrix such that at each step, tighter system availability bounds can be obtained. One important note is that the algorithm can be terminated at any phase depending on the tightness of bounds required. This is due to the fact that at all times, we have bounds on the system availability.

3.7 Partial state generation

In the previous sections, we developed an algorithm to compute bounds on the steady state availability bounds of repairable computer systems. The algorithm assumes a rate matrix generation process in which it is feasible to generate portions of the matrix in chunks which correspond to sets of states of the form $\bigcup_{i=L}^{H} \mathcal{F}_i$ for some L and H. The problem is that $|\mathcal{F}_i|$ can be too large to be handled in one step (e.g., model of a system with 100 components, $|\mathcal{F}_{10}| > 10^9$. When this occurs, in one step of the computation procedure, $\mathcal{D}$ will have to be restricted to a proper subset of states in $\mathcal{F}_i$ for some i.). In this section, we extend the algorithm so that it can accommodate this generality.

Assume we want to compute lower bounds for the state probabilities of states in $\{\mathcal{F}_0 \cup \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_k\}$ using the one-step bounding algorithm. Let G in Figure 3.1 be the original transition rate matrix. After applying cloning to all states in $\{\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_k\}$, we obtain the transition rate matrix G_1 as depicted in Figure 3.11.

Assume that due to memory limitations, we can handle the portion of the rate matrix corresponding to $\mathcal{D} = \left\{\mathcal{F}_0 \cup \cdots \mathcal{F}_{k-1} \cup \mathcal{F}_k^{'}\right\}$, where $\mathcal{F}_k^{'}$ is a proper subset of $\mathcal{F}_k$. Let $\mathcal{F}_k^{''} = \mathcal{F}_k - \mathcal{F}_k^{'}$ The corresponding transition rate matrix G_2 is depicted in Figure 3.12.

In order to compute lower bounds for the state probabilities of states in the set $\mathcal{D}$, we transform the rate matrix G_2 to G_3 (depicted in Figure 3.13). The transformation has the following probabilistic interpretation. Assume initially the system is in one of the states in $\mathcal{D}$, whenever there is a transition from state i in $\mathcal{D}$ to state j in $\mathcal{F}_k^{''}$, this becomes a transition from state i in $\mathcal{D}$ to state j in $\mathcal{C}_k^{''}$ instead. Since we apply cloning to states in $\{\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_k\}$, we guarantee there is a one-to-one mapping to its clone state.

Let $\pi_{\mathcal{D}/G_i}$ be the vector of stationary probabilities of states in $\mathcal{D}$ with rate matrix G_i. The following theorem indicates that the solution of the transformed matrix provides lower bounds for the state probabilities of states in $\mathcal{D}$.

Theorem 3.5 $\pi_{\mathcal{D}/G_3} \leq \pi_{\mathcal{D}/G_2}$

Proof : The proof is given in the Appendix. □

We can now transform G_3 to G_4 (depicted in Figure 3.14) by exact aggregation. We aggregate all the states in $\mathcal{F}_i$ for $i > k$ and all clone states according to number of failed components. Since we apply exact aggregation, we have $\pi_{\mathcal{D}/G_3} = \pi_{\mathcal{D}/G_4}$.

At this point, we can apply the theorem in Section 3.3 and replace the aggregate transition rates by bounds. The final transition rate matrix, G_5 is given in Figure 3.15. It is easy to see that $\pi_{\mathcal{D}/G_5} \leq \pi_{\mathcal{D}/G}$.

With this modification, we overcome the partial state generation problem and we also maintain the block Hessenberg property.

For the multi-step bounding algorithm, we may encounter the following two situations:

1. we can generate all the $\mathcal{F}_k''$ from the previous step and perhaps states with more failures.

2. we can only generate a proper subset of $\mathcal{F}_k''$.

We can apply the same transformations we used in the one-step bounding algorithm to handle these cases. The modifications are:

1. Let $\mathcal{F}_k''$ be the those states in $\mathcal{F}_k$ which are not in $\mathcal{D}$. All transitions from $\mathcal{D}$ to any states in $\mathcal{F}_k''$ are changed to transitions to the corresponding clone states.

2. Apply upper and lower bounds transition rates for the aggregate and the clone states.

It is easy to show that this modification provides a transition rate matrix whose solution vector provides the lower bound steady state probabilities for the states in $\mathcal{D}$.

For the bound spread reduction process, we see that to refine the errors for clone states that have k or less failures, we have to have lower bounds for the state probabilities for all the states with $k+1$ failures. We can obtain lower bounds on the state probabilities for the states with $k+1$ failures using the multi-step algorithm. Once we get the lower bounds on the state probabilities for the states with $k+1$ failures, we can apply the iterative refinement process as described in Section 3.5.

$$\left[\begin{array}{ccccc|cccc|ccc}
Q_{00} & Q_{01} & \cdot\cdot & Q_{0,k-1} & Q_{0,k} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{0,n-1} & Q_{0,n} \\
Q_{10} & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & Q_{k,k-1} & Q_{k,k} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{k,n-1} & Q_{k,n} \\
\hline
Q_{10} & 0 & \cdot\cdot & 0 & 0 & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & 0 & \cdot\cdot & 0 & 0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k,k-1} & Q_{k,k} & \cdot\cdot & Q_{k,n-1} & Q_{k,n} \\
\hline
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & Q_{k+1,k} & \cdot\cdot & Q_{k+1,n-1} & Q_{k+1,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{n,n-1} & Q_{n,n}
\end{array}\right]$$

Figure 3.11: Transition matrix G_1.

$$\left[\begin{array}{ccccc|c|ccccc|ccc}
Q_{00} & Q_{01} & \cdot\cdot & Q_{0,k-1} & Q_{0,k'} & Q_{0,k''} & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{0,n-1} & Q_{0,n} \\
Q_{10} & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & Q_{1,k''} & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & Q_{2,k''} & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
\hline
0 & 0 & \cdot\cdot & Q_{k'',k-1} & 0 & Q_{k'',k''} & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k'',n-1} & Q_{k'',n} \\
Q_{10} & 0 & \cdot\cdot & 0 & 0 & 0 & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & Q_{1,k''} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & Q_{2,k''} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & 0 & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & 0 & \cdot\cdot & Q_{k'',k-1} & 0 & Q_{k'',k''} & \cdot\cdot & Q_{k'',n-1} & Q_{k'',n} \\
\hline
0 & 0 & \cdot\cdot & 0 & 0 & 0 & 0 & \cdot\cdot & 0 & Q_{k+1,k'} & Q_{k+1,k''} & \cdot\cdot & Q_{k+1,n-1} & Q_{k+1,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{n,n-1} & Q_{n,n}
\end{array}\right]$$

Figure 3.12: Transition matrix G_2.

$$\left[\begin{array}{ccccc|ccccc|ccc}
Q_{00} & Q_{01} & \cdot\cdot & Q_{0,k-1} & Q_{0,k'} & 0 & \cdot\cdot & 0 & 0 & Q_{0,k''} & \cdot\cdot & Q_{0,n-1} & Q_{0,n} \\
Q_{10} & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & 0 & \cdot\cdot & 0 & 0 & Q_{1,k''} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & 0 & \cdot\cdot & 0 & 0 & Q_{2,k''} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
\hline
Q_{10} & 0 & \cdot\cdot & 0 & 0 & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & Q_{1,k''} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & 0 & \cdot\cdot & 0 & 0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & Q_{2,k''} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{k'',k-1} & 0 & Q_{k'',k''} & \cdot\cdot & Q_{k'',n-1} & Q_{k'',n} \\
\hline
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & Q_{k+1,k'} & Q_{k+1,k''} & \cdot\cdot & Q_{k+1,n-1} & Q_{k+1,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & Q_{n,n-1} & Q_{n,n}
\end{array}\right]$$

Figure 3.13: Transition matrix G_3.

$$\left[\begin{array}{ccccc|cccc|ccc}
Q_{00} & Q_{01} & \cdot\cdot & Q_{0,k-1} & Q_{0,k'} & 0 & \cdot\cdot & 0 & Q_{0,k''} & \cdot\cdot & Q_{0,n-1} & Q_{0,n} \\
Q_{10} & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & 0 & \cdot\cdot & 0 & Q_{1,k''} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & 0 & \cdot\cdot & 0 & Q_{2,k''} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
\hline
r_{10} & 0 & \cdot\cdot & 0 & 0 & \bullet & \cdot\cdot & r_{1,k-1} & r_{1,k} & \cdot\cdot & r_{1,n-1} & r_{1,n} \\
0 & 0 & \cdot\cdot & 0 & 0 & r_{21} & \cdot\cdot & r_{2,k-1} & r_{2,k} & \cdot\cdot & r_{2,n-1} & r_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & r_{k,k-1} & \bullet & \cdot\cdot & r_{k',n-1} & r_{k,n} \\
\hline
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & r_{k+1,k} & \cdot\cdot & r_{k+1,n-1} & r_{k+1,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & r_{n,n-1} & \bullet
\end{array}\right]$$

Figure 3.14: Transition matrix G_4.

$$\left[\begin{array}{ccccc|cccc|ccc}
Q_{00} & Q_{01} & \cdot\cdot & Q_{0,k-1} & Q_{0,k'} & 0 & \cdot\cdot & 0 & Q_{0,k''} & \cdot\cdot & Q_{0,n-1} & Q_{0,n} \\
Q_{10} & Q_{11} & \cdot\cdot & Q_{1,k-1} & Q_{1,k'} & 0 & \cdot\cdot & 0 & Q_{1,k''} & \cdot\cdot & Q_{1,n-1} & Q_{1,n} \\
0 & Q_{21} & \cdot\cdot & Q_{2,k-1} & Q_{2,k'} & 0 & \cdot\cdot & 0 & Q_{2,k''} & \cdot\cdot & Q_{2,n-1} & Q_{2,n} \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & Q_{k',k-1} & Q_{k',k'} & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & Q_{k',n-1} & Q_{k',n} \\
\hline
- & 0 & \cdot\cdot & 0 & 0 & \bullet & \cdot\cdot & + & + & \cdot\cdot & + & + \\
0 & 0 & \cdot\cdot & 0 & 0 & - & \cdot\cdot & + & + & \cdot\cdot & + & + \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & - & \bullet & \cdot\cdot & + & + \\
\hline
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & - & \cdot\cdot & + & + \\
\vdots & \vdots & \cdot\cdot & \vdots & \vdots & \vdots & \cdot\cdot & \vdots & \vdots & \cdot\cdot & \vdots & \vdots \\
0 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 & \cdot\cdot & - & \bullet
\end{array}\right]$$

Figure 3.15: Transition matrix G_5.

3.8 Example.

In this section, we will present an example to illustrate the application of the bounding algorithm. This example incorporates the general bounding procedure and a tight bound is obtained.

The example is a heterogeneous distributed database system as depicted in Figure 3.16. The components of this system are: two front-ends, four databases and four processing subsystems consisting of a switch, a memory and two processors. Components may fail and be repaired according to the rates given in Table 3.1. If either processors of subsystem A or B fail, they have a 0.05 probability of contaminating both the database A and a. If either processors of subsystem C or D fail, they have a 0.05 probability of contaminating the database B and b. Components are repaired by a single repair facility which gives preemptive priority to components in the order: front-end, databases, switches, memories, processors set 1 and lastly processor set 2. (Ties are broken by random selection.) The database system is considered operational if at least one front-end is operational, at least one database is operational, and at least one processing subsystem is operational. A processing subsystem is operational if the switch, the memory and at least one processor are operational. Also, this system is in *active breakdown mode*, meaning that components fail even when the system is non-operational.

In Table 3.2, we present the bounds on steady state availability for several steps of the bounding procedure. We note that for each step, the bounds are significantly tightened. In step one, we apply the one-step bounding algorithm with detail states that have 0 to 2 failed components. In step two, we apply the multi-step bounding algorithm for detailed states that have between 3 and 4

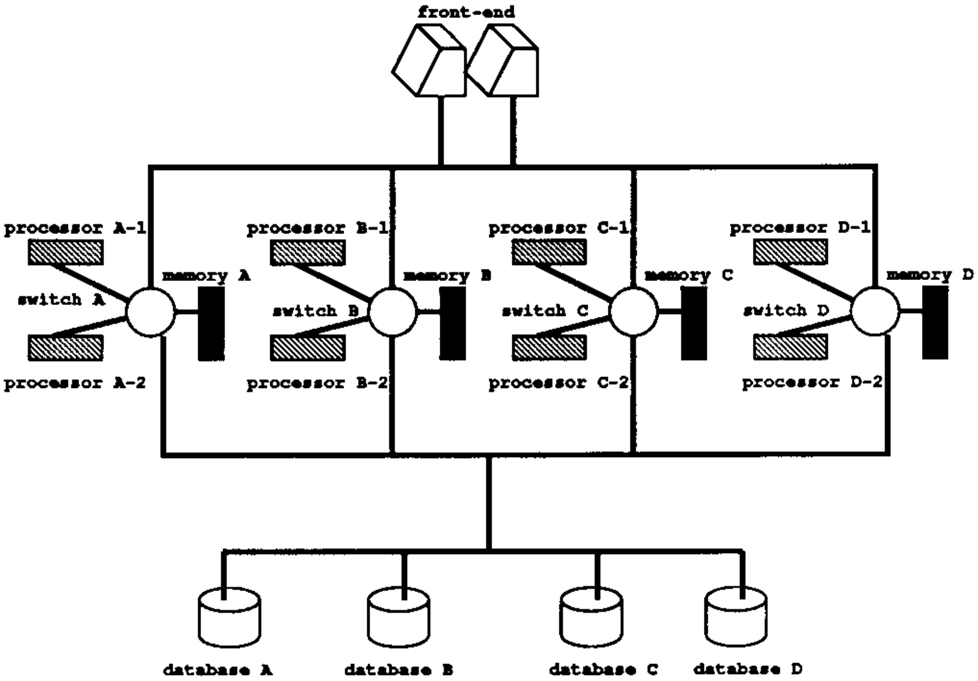

Figure 3.16: A fault-tolerant heterogeneous distributed database system.

failed components. In step three, we apply the bound spread reduction algorithm described in Section 3.5.3 for states that have between 1 and 4 failed components. In step four, we apply the multi-step bounding algorithm with detailed states that have between 5 and 6 failed components. In step five, we apply the bound spread reduction algorithm for states that have between 1 and 6 failed components.

In Table 3.3 and Table 3.4, we illustrate the individual contributions to the bound spread reduction by the clone states and detailed states when the bound spread reduction algorithm is applied. These data show that most of the contribution is from the unclaimed reward of the clone states. In fact, the majority of the gain comes from obtaining lower bounds on the clone states for which the bound spread reduction is applied for the first time.

Table 3.5 shows details of the reduction in the spread of the availability bounds in step 3 of the global algorithm. The first two columns show the estimated gain in each direction, the third and fourth columns show the spread in the availability bounds if we follow the decision algorithm. The last two columns show the spread in the availability bounds if the decision algorithm is not followed. In the first row, we observe that by applying the bound spread reduction algorithm, the spread of the availability bounds is significantly reduced compared to the spread of the availability bounds if the multi-step bounding (or forward) algorithm is applied. In the second row, since the estimated gain g_b' is greater than g_f', we apply the bound spread reduction algorithm again. The reduction in the spread of the availability bounds is comparable in either direction, but since the cost of forward generation is much more expensive, it pays to apply the bound spread reduction algorithm. In the last row (which corresponds to the step 4 of the global algorithm), since the estimated gain g_f' is greater than g_b', we apply the multi-step bounding algorithm and we obtain a significant reduction in the spread of the availability bounds. One important note about the decision algorithm is that it cannot get in an infinite loop in applying the bound spread reduction algorithm because as defined in Equation (3.21), g_b' will eventually go to zero.

Components	Mean Failure Rate	Mean Repair Rate
Front-end A	1/4000	2.1
Front-end B	1/8000	2.0
Processor A-1	1/500	2.5
Processor A-2	1/400	2.0
Switch A	1/750	2.7
Memory A	1/750	2.5
Processor B-1	1/450	2.3
Processor B-2	1/450	1.8
Switch B	1/625	2.6
Memory B	1/750	2.4
Processor C-1	1/600	2.3
Processor C-2	1/450	1.7
Switch C	1/625	2.6
Memory C	1/600	2.4
Processor D-1	1/450	2.1
Processor D-2	1/450	1.5
Switch C	1/600	2.1
Memory C	1/600	2.5
Database A	1/5500	2.5
Database a	1/5000	2.2
Database B	1/5000	2.5
Database b	1/4500	2.3

Table 3.1: Failure and repair rates(per hour).

Step	Algorithm	Availability	Spread in
Number	applied	(Upper Bound	Availability
		lower Bound)	Bounds
1	one-step	0.986456955373	0.013543044282
		0.999999999655	
2	multi-step	0.990023127431	0.009976869598
	(one application)	0.999999999029	
3	bound spread reduction	0.999995763421	0.000004224222
	(applied twice)	0.999999987643	
4	multi-step	0.999999246581	0.000000728630
	(one application)	0.999999975211	
5	bound spread reduction	0.999999952345	0.000000000627
	(applied twice)	0.999999952972	

Table 3.2: Upper and lower bounds on steady state availability of the database system.

total reduction of bound spread for step 3	9.9726 x 10^{-3}
contribution by clone states $\mathcal{C}_1 \cup \mathcal{C}_2$	8.2304 x 10^{-3}
contribution by detailed states $\mathcal{D}_0 \cup \cdots \cup \mathcal{D}_4$	1.7422 x 10^{-3}

Table 3.3: Contribution by clones states and detail states for step 3

total reduction of bound spread for step 5	7.2800 x 10^{-7}
contribution by clone states $\mathcal{C}_1 \cup \mathcal{C}_2$	5.0818 x 10^{-8}
contribution by clone states $\mathcal{C}_3 \cup \mathcal{C}_4$	5.8400 x 10^{-7}
contribution by detailed states $\mathcal{D}_0 \cup \cdots \cup \mathcal{D}_2$	3.5391 x 10^{-9}
contribution by detailed states $\mathcal{D}_3 \cup \mathcal{D}_4$	7.3522 x 10^{-8}
contribution by detailed states $\mathcal{D}_5 \cup \mathcal{D}_6$	1.6122 x 10^{-8}

Table 3.4: Contribution by clones states and detail states for step 5

g_f'	g_b'	algorithm applied	resulting spread in availability bounds	algorithm applied	resulting spread in availability bounds
		bound spread reduction	5.6821x10^{-6}	multi-step bounding	9.9764x10^{-3}
1.4962x10^{-6}	5.6788x10^{-6}	bound spread reduction	4.2242x10^{-6}	multi-step bounding	4.4781x10^{-6}
1.1126x10^{-6}	1.0838x10^{-6}	multi-step reduction	7.2863x10^{-7}	bound spread reduction	3.7822x10^{-6}

Table 3.5: Illustration for decision algorithm

3.9 Conclusions.

In this chapter, we have developed a methodology for computing bounds on the steady state availability of repairable computer systems. The method provides an efficient way to overcome the large state space problem in evaluating steady state availability of realistic systems. We showed that by modification of the original model, bounds can be obtained with much less cost and also state space cardinality was drastically reduced. Depending on the tightness required, the user can tradeoff tightness of the bounds with computational effort.

The development in the chapter is couched in terms of models of repairable computer systems and determining bounds on availability. However the methods appear to have promise for other applications. The important property of availability models that was used was that the equilibrium state probabilities were concentrated in very few states. It is reasonable to expect that this same property will hold for example, in models of probabilistic protocol evaluation [DC88, MS87] and load balancing. In the case of load balancing, there is presumably a policy for balancing the load on the resources in the system. Thus we expect that a large number of possible states of the system will have "small" probability since the scheduler will be biasing the system toward a small number of preferred states.

CHAPTER 4

GENERALIZATION FOR AVAILABILITY BOUNDING METHODOLOGY

In Chapter 3, presented an algorithm to bound steady state availability based on a Markov model of a system under study. To obtain these results, an assumption was made that the transition rate matrix had a block upper Hessenberg form. In this chapter, we extend these results to a Markov process whose rate matrix has a general transition structure. We assume the original Markov process has very large state space and we can only generate the portion of the rate matrix corresponding to transition state 0 thru state I (this assumption may be due to memory limitations for example). The objective is to bound the steady state expected performance measure, e.g, steady state availability, given only the information from the northwest corner of the transition rate matrix (transition rates between state 0 to state I). Since the given transition rate matrix has general structure, we do not guarantee that we obtain tight bounds in all situations. We also discuss under what situations this bounding procedure will give an acceptable performance bounds.

4.1 Problem Definition

Given a continuous time homogeneous Markov process with a transition rate matrix as follows:

$$G = \left[\begin{array}{cccc|ccc} q_{0,0} & q_{1,1} & \cdots & q_{0,I} & q_{0,I+1} & \cdots & q_{0,J} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,I} & q_{1,I+1} & \cdots & q_{1,J} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ q_{I,0} & q_{I,1} & \cdots & q_{I,I} & q_{I,I+1} & \cdots & q_{I,J} \\ \hline q_{I+1,0} & q_{I+1,1} & \cdots & q_{I+1,I} & q_{I+1,I+1} & \cdots & q_{I+1,J} \\ q_{I+2,0} & q_{I+2,1} & \cdots & q_{I+2,I} & q_{I+2,I+1} & \cdots & q_{I+2,J} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ q_{J,0} & q_{J,1} & \cdots & q_{J,I} & q_{J,I+1} & \cdots & q_{J,J} \end{array}\right]$$

We assume that we can only generate the transition structure of the original Markov process from state 0 to state I (the northwest corner of the rate matrix). Once we only know the transition rate submatrix, the objective is to bound the steady state performance measure of the system, e.g., steady state availability. We make the following assumptions about our model:

1. The underlying Markov process is irreducible.

2. The state space of the northwest corner of the transition rate matrix is large.

3. The transition rate matrix of the northwest corner is sparse and has an irregular transition structure, for example, it is not a diagonal rate matrix.

The reason for having large state space, sparsity and irregular transition structure is that performing a formal inverse on the northwest corner of the transition rate matrix is computational prohibitive.

We approach this problem by modifying the transition structure outside the northwest corner, and apply aggregation techniques to reduce the state space of the problem and show that by bounding the aggregate transition rates, we can obtain error bounds.

4.2 Approach

Let us construct another Markov process with transition rate matrix $\hat{G}$:

$$\begin{array}{lcll}
\hat{q}_{i,j} & = & q_{i,j} & i = 0, 1, \ldots, I \quad j = 0, 1, \ldots, J \\
\hat{q}_{i,J+1} & = & 0 & i = 0, 1, \ldots, I \\
\hat{q}_{i,j} & = & 0 & i = I+1, \ldots, J, \quad j = 0, 1, \ldots, I \\
\hat{q}_{i,j} & = & q_{i,j} & i = I+1, \ldots, J, \quad j = I+1, \ldots, J \\
\hat{q}_{i,J+1} & = & \sum_{j=0}^{I} q_{i,j} & i = I+1, \ldots, J \\
\hat{q}_{J+1,j} & = & \left[\sum_{i=I+1}^{J} \pi_i q_{i,j}\right] / (\sum_{i=I+1}^{J} \pi_i) & j = 0, 1, \ldots, I \\
\hat{q}_{J+1,j} & = & 0 & j = I+1, \ldots, J \\
\hat{q}_{J+1,J+1} & = & -\sum_{j=0}^{I} \hat{q}_{J+1,j} &
\end{array}$$

where π_i is the steady state probability of state i in the original model.

The transition rate matrix for the modified process is:

$$\hat{G} = \left[\begin{array}{cccc|ccc|c}
\hat{q}_{0,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{0,I} & \hat{q}_{0,I+1} & \cdots & \hat{q}_{0,J} & 0 \\
\hat{q}_{1,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{1,I} & \hat{q}_{1,I+1} & \cdots & \hat{q}_{1,J} & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & 0 \\
\hat{q}_{I,0} & \hat{q}_{I,1} & \cdots & \hat{q}_{I,I} & \hat{q}_{I,I+1} & \cdots & \hat{q}_{I,J} & 0 \\
\hline
0 & 0 & \cdots & 0 & \hat{q}_{I+1,I+1} & \cdots & \hat{q}_{I+1,J} & \sum_{j=0}^{I} q_{I+1,j} \\
0 & 0 & \cdots & 0 & \hat{q}_{I+2,I+1} & \cdots & \hat{q}_{I+2,J} & \sum_{j=0}^{I} q_{I+2,j} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \hat{q}_{J,I+1} & \cdots & \hat{q}_{J,J} & \sum_{j=0}^{I} q_{J,j} \\
\hline
\hat{q}_{J+1,0} & \hat{q}_{J+1,1} & \cdots & \hat{q}_{J+1,I} & 0 & \cdots & 0 & \hat{q}_{J+1,J+1}
\end{array}\right]$$

Let $\hat{\pi}_i$ be the steady state probability of state i in rate matrix $\hat{G}$. We can establish the following theorem:

Theorem 4.1

$$\begin{aligned}
\pi_i &= \frac{\hat{\pi}_i}{\sum_{k=0}^{J} \hat{\pi}_k} \quad i = 0, 1, \ldots, J \quad iff \\
\sum_{i=I+1}^{J} \hat{\pi}_i(q_{i,j}) &= \hat{\pi}_{J+1}\hat{q}_{J+1,j} \quad for\ j = 0, 1, \ldots, I
\end{aligned}$$

Proof: We first prove the ($\Rightarrow$) part. From the balance equation for state $J+1$, we have:

$$\sum_{k=I+1}^{J} \hat{\pi}_k \hat{q}_{k,J+1} = \hat{\pi}_{J+1} \sum_{j=0}^{I} \hat{q}_{J+1,j} \tag{4.1}$$

By construction, we have:

$$\sum_{k=I+1}^{J} \hat{\pi}_k (\sum_{j=0}^{I} q_{k,j}) = \frac{\hat{\pi}_{J+1}}{\sum_{i=I+1}^{J} \pi_i} \left(\sum_{j=0}^{I} \sum_{k=I+1}^{J} \pi_k q_{k,j} \right) \tag{4.2}$$

By rearranging terms, we have:

$$\sum_{j=0}^{I}\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \frac{\hat{\pi}_{J+1}}{\sum_{i=I+1}^{J} \pi_i}\left(\sum_{j=0}^{I}\sum_{k=I+1}^{J} \pi_k q_{k,j}\right) \tag{4.3}$$

Since:

$$\pi_i \quad = \quad \frac{\hat{\pi}_i}{\sum_{k=0}^{J} \hat{\pi}_k} \qquad \mathit{for}\ i = 0, 1, \ldots, J$$

Substituting them into Equation 4.3, we have:

$$\sum_{j=0}^{I}\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \frac{\hat{\pi}_{J+1}}{\sum_{i=I+1}^{J} \hat{\pi}_i}\left(\sum_{j=0}^{I}\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j}\right) \tag{4.4}$$

Canceling $\sum_{j=0}^{I}\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,J+1}$ from both sides of the equation, we then have:

$$\hat{\pi}_{J+1} \quad = \quad \sum_{k=I+1}^{J} \hat{\pi}_k \tag{4.5}$$

Now, let us examine the balance equation of state $J+1$. This equation will hold if each of the individual local balance equations holds, namely:

$$\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \hat{\pi}_{J+1}\hat{q}_{J+1,j} \quad \mathit{for}\ j = 0, 1, \ldots, I,$$

If we expand any one of the above equations, we have:

$$\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \frac{\hat{\pi}_{J+1}}{\sum_{i=I+1}^{J} \hat{\pi}_i}\left(\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j}\right) \quad \mathit{for}\ j = 0, 1, \ldots, I.$$

But since Equation 4.5 holds, it follows that each of the individual local balance equations:

$$\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \hat{\pi}_{J+1}\hat{q}_{J+1,j} \quad \mathit{for}\ j = 0, 1, \ldots, I.$$

holds.

For the ($\Leftarrow$) part. If we look at the balance equations for state 0 through state I, we have:

$$\sum_{i=0}^{I} \hat{\pi}_i q_{i,j} + \hat{\pi}_{J+1}\hat{q}_{J+1,j} \quad = \quad 0 \quad \mathit{for}\ j = 0, 1, \ldots, I \tag{4.6}$$

Since:

$$\sum_{k=I+1}^{J} \hat{\pi}_k q_{k,j} \quad = \quad \hat{\pi}_{J+1} \hat{q}_{J+1,j} \;\; for \; j = 0, 1, \ldots, I \tag{4.7}$$

Combining the equations above with the balance equations for states $I+1$ to J, we have:

$$\sum_{i=0}^{I} \hat{\pi}_i q_{i,j} + \sum_{i=I+1}^{J} \hat{\pi}_i q_{i,j} \quad = \quad 0 \;\; for \; j = 0, 1, \ldots, J \tag{4.8}$$

Since these balance equations for states 0 to state J have the same form as the balance equations for state 0 to state J in rate matrix G, we conclude that:

$$C\hat{\pi}_i \quad = \quad \pi_i, \qquad\qquad i = 0, 1, \ldots, J$$

where C is a constant. Summing over 0 to J, we have:

$$\sum_{i=0}^{J} \pi_i \quad = \quad 1$$

therefore:

$$\begin{aligned} C &= (\sum_{k=0}^{J} \hat{\pi}_k)^{-1} \qquad\qquad and \\ \pi_i &= \hat{\pi}_i / (\sum_{k=0}^{J} \hat{\pi}_k) \qquad for \; i = 0, 1, \ldots, J \qquad\qquad \square \end{aligned}$$

For the remainder of this chapter, let $\hat{\pi}_{i|E} = \hat{\pi}_i / (\sum_{k=0}^{J} \hat{\pi}_k)$ to simply the notation.

Based on Theorem 4.1, we know that if we can bound $\hat{\pi}_{i|E}$, we can bound π_i. In the following, we derive a bound for $\hat{\pi}_{i|E}$. Let us construct the following stochastic matrices $\hat{G}'_k, k = 0, 1, \ldots, I$ from $\hat{G}$, namely:

$$\begin{array}{llll} \hat{q}'_{i,j} & = & \hat{q}_{i,j} & i = 0, 1, \ldots, I \; j = 0, 1, \ldots, J \\ \hat{q}'_{i,j} & = & 0 & i = I+1, \ldots, J, \; j = 0, 1, \ldots, k-1, k+1, \ldots, I \\ \hat{q}'_{i,k} & = & \hat{q}_{i,J+1} \; = \sum_{j=0}^{I} q_{i,j} & i = I+1, \ldots, J \\ \hat{q}'_{i,j} & = & \hat{q}_{i,j} & i = I+1, \ldots, J, \; j = I+1, \ldots, J \end{array}$$

$$\hat{G}_0' = \left[\begin{array}{cccc|ccc}
\hat{q}_{0,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{0,I} & \hat{q}_{0,I+1} & \cdots & \hat{q}_{0,J} \\
\hat{q}_{1,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{1,I} & \hat{q}_{1,I+1} & \cdots & \hat{q}_{1,J} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\hat{q}_{I,0} & \hat{q}_{I,1} & \cdots & \hat{q}_{I,I} & \hat{q}_{I,I+1} & \cdots & \hat{q}_{I,J} \\
\hline
\sum_{j=0}^{I} q_{I+1,j} & 0 & \cdots & 0 & \hat{q}_{I+1,I+1} & \cdots & \hat{q}_{I+1,J} \\
\sum_{j=0}^{I} q_{I+2,j} & 0 & \cdots & 0 & \hat{q}_{I+2,I+1} & \cdots & \hat{q}_{I+2,J} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\sum_{j=0}^{I} q_{J,j} & 0 & \cdots & 0 & \hat{q}_{J,I+1} & \cdots & \hat{q}_{J,J}
\end{array}\right]$$

Figure 4.1: rate matrix $\hat{G}_0'$

Figure 4.2 illustrates a particular $\hat{G}_k'$, namely, rate matrix $\hat{G}_0'$.

Let us define z_i to be the steady state probability vector for rate matrix $\hat{G}_i'$, $i = 0, 1, \ldots, I$. We can establish the following theorem:

Theorem 4.2

$$\hat{\pi}_{i|E} = \sum_{i=0}^{I} \beta_i z_i \qquad where \quad \sum_{i=0}^{I} \beta_i = 1$$

Proof: Based on Courtois' result in [CS86]. □

From the above theorem, we can bound each individual state probability for states $i = 0, 1, \ldots, I$. But since we only know the transition structure in the northwest corner, namely transitions between state 0 to state I. We do not know other transition rates in matrix $\hat{G}_i', i = 0, 1, \ldots, I$. In the following, we will construct a new matrix $\hat{G}_i''$ for each $i = 0, 1, \ldots, I$ and bound the steady state probability vector z_i.

Let the transition rate in $\hat{G}_k''$ be as follows:

$$\begin{array}{lcll}
\hat{q}_{i,j}'' & = & \hat{q}_{i,j}' & i = 0, 1, \ldots, I \quad j = 0, 1, \ldots, J \\
\hat{q}_{I+1,k}'' & = & \hat{q}_{I+1,k}' & \\
\hat{q}_{I+1,j}'' & = & \hat{q}_{I+1,j}' & j = I+1, I+2, \ldots, J \\
\hat{q}_{i,i-1}'' & = & \hat{q}_{i,k}' + \sum_{k=I+1}^{i-1} \hat{q}_{i,k} & i = I+2, I+3, \ldots, J \\
\hat{q}_{i,j}'' & = & \hat{q}_{i,j}' & i = I+2, I+3, \ldots, J \;\; j \geq i \\
\hat{q}_{i,j}'' & = & 0 & otherwise
\end{array}$$

Figure 4.2 illustrate $\hat{G}_0''$.

$$\hat{G}_0'' = \left[\begin{array}{cccc|cccc}
\hat{q}_{0,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{0,I} & \hat{q}_{0,I+1} & \hat{q}_{0,I+2} & \cdots & \hat{q}_{0,J} \\
\hat{q}_{1,0} & \hat{q}_{1,1} & \cdots & \hat{q}_{1,I} & \hat{q}_{1,I+1} & \hat{q}_{1,I+2} & \cdots & \hat{q}_{1,J} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
\hat{q}_{I,0} & \hat{q}_{I,1} & \cdots & \hat{q}_{I,I} & \hat{q}_{I,I+1} & \hat{q}_{I,I+2} & \cdots & \hat{q}_{I,J} \\
\hline
\sum_{j=0}^{I} q_{I+1,j} & 0 & \cdots & 0 & \hat{q}_{I+1,I+1} & \hat{q}_{I+1,I+2} & \cdots & \hat{q}_{I+1,J} \\
0 & 0 & \cdots & 0 & \sum_{j=0}^{I+1} q_{I+2,j} & \hat{q}_{I+2,I+2} & \cdots & \hat{q}_{I+2,J} \\
0 & 0 & \cdots & 0 & 0 & \sum_{j=0}^{I+2} q_{I+3,j} & \cdots & \hat{q}_{I+3,J} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & \hat{q}_{J,J}
\end{array}\right]$$

Figure 4.2: rate matrix $\hat{G}_0''$

Let $\pi_{i|G}$ be the steady state probability of state i in rate matrix G. We can show the following:

Theorem 4.3

$$\pi_{i|\hat{G}_k'} \geq \pi_{i|\hat{G}_k''} \qquad for\ i,k = 0,1,\ldots I$$

Proof : Given that G is a transition rate matrix for a finite, irreducible continuous time Markov chain $\mathcal{G}$. Then $\mathcal{G}$ is uniformizable which means that $\mathcal{G}$ can be transformed to a discrete time Markov chain with transition probability matrix P which has the same stationary probability vector π. This transformation is achieved by:

$$P = I + \lambda^{-1} G$$

where λ is greater than or equal to the largest absolute diagonal element of G. Here, we uniformize the two rate matrices $\hat{G}_k'$ and $\hat{G}_k''$ into P_k' and P_k'' with $\lambda = \max(\lambda_k', \lambda_k'')$ where λ_k' (λ_k'') is the largest of the absolute values of the diagonal elements of $\hat{G}_k'$ $(\hat{G}_k'')$.

We define the following notation:

$p_{i,j}'$ = the transition probability from state i to state j in P_k'.

$p_{i,j}''$ = the transition probability from state i to state j in P_k''.

$r(i)$ = reward (either 0 or 1) for state i.

$T_k'[r(i)]$ = one-step expected reward of P_k' given the present state is i, i.e.

$$T_k'[r(i)] = \sum_j p_{ij}' \, r(j)$$

$T_k''[r(i)]$ = one-step expected reward of P_k'' given the present state is i, i.e.

$$T_k''[r(i)] = \sum_j p_{ij}'' \, r(j)$$

$R_k^{'n}(i)$ = n-step ($n-1$ transitions plus initial position) accumulative reward for $P_k^{'}$ given the initial state is i, i.e.

$$R_k^{'n}(i) \;=\; \sum_{l=0}^{n-1} T_k^{'l}[r(i)] \qquad\qquad for\ n = 1,2,3,\ldots$$

and $T_k^{'0}[r(i)]$ be an identity function.

$R_k^{''n}(i)$ = n-step ($n-1$ transitions plus initial position) accumulative reward for $P_k^{''}$ given the initial state is i.

$1\{c\}$ = an indicator function equal to 1 if the condition c is true, else 0.

In order to prove the theorem, it is sufficient to show that:

$$(T_k^{'} - T_k^{''})R_k^{'n}(i) \;\geq\; 0 \qquad\qquad \forall i\ and\ \forall n$$

Let f be any nonnegative function applied to state i of the Markov chain. For any nonnegative function f:

$$(T_k^{'} - T_k^{''})f(i) \;=\; \sum_{j=I+2}^{J} 1\{i=j\}\left\{p_{j,k}^{'}f(k) + \sum_{l=I+1}^{J} p_{j,l}^{'}f(l) - \sum_{l=j-1}^{J} p_{j,l}^{''}f(l)\right\}$$

Since:

$$\begin{aligned} p_{j,j-1}^{''} &= p_{j,k}^{'} + \textstyle\sum_{l=I+1}^{j-1} p_{j,l}^{'} && j = I+2,\ldots,J \\ p_{j,l}^{''} &= p_{j,l}^{'} && j = I+2,\ldots,J \quad l = j, j+1,\ldots,J \end{aligned}$$

it follows that:

$$(T_k^{'} - T_k^{''})f(i) = \sum_{j=I+2}^{J} 1\{i=j\}\left\{p_{j,k}^{'}[f(k)-f(j-1)] + \sum_{l=I+1}^{j-1} p_{j,l}^{'}[f(l)-f(j-1)]\right\}$$

The $(T_k^{'} - T_k^{''})f(i)$ will be greater than 0 if the following conditions are satisfied:

$$\begin{aligned} f(i) &\geq f(j) && i = 0,1,\ldots,I\ and\ j = I+1, I+2,\ldots,J \\ f(j) &\geq f(j+1) && j = I+1, I+2,\ldots,J-1 \end{aligned}$$

Let f be $R_k^{'n}(i)$, then to prove the theorem, we have to show that the following conditions are true:

$$\begin{array}{lll} R_k^{'n}(i) - R_k^{'n}(j) & \geq 0 & i = 0,1,\ldots,I \;\; j = I+1, I+2, \ldots, J \;\; \forall n \\ R_k^{'n}(j) - R_k^{'n}(j+1) & \geq 0 & j = I+1, I+2, \ldots, J-1 \;\; \forall n \end{array}$$

To prove the above inequalities, we assign the reward function for each state as follow:

$$r(i) = \begin{cases} 1 & 0 \leq i \leq I \\ 0 & otherwise \end{cases}$$

Now we can prove the above inequalities by induction. Since the approach is similar in each case, we only give the details for the case where $i = k$ and $j = I+1$.

For $n = 0$, since $R_k^{'0}(i) = 0$ for any state i, the inequality holds.

Assume the inequality holds for n, we need to show:

$$\begin{aligned} R_k^{'n+1}(k) - R_k^{'n+1}(I+1) &= R_k^{'n+1}(k) - \Big\{ r(I+1) + p'_{I+1,k} R_k^{'n}(k) \\ &\qquad + \textstyle\sum_{l=I+1}^{J} p'_{I+1,l} R_k^{'n}(l) \Big\} \\ &= R_k^{'n+1}(k) - \Big\{ p'_{I+1,k} R_k^{'n}(k) + \textstyle\sum_{l=I+1}^{J} p'_{I+1,l} R_k^{'n}(l) \Big\} \\ &\geq R_k^{'n+1}(k) - \Big\{ p'_{I+1,k} R_k^{'n}(k) + \textstyle\sum_{l=I+1}^{J} p'_{I+1,l} R_k^{'n}(k) \Big\} \\ &= R_k^{'n+1}(k) - R_k^{'n}(k) \\ &\geq 0 \qquad\qquad \Box \end{aligned}$$

Now, given all rate matrices $\hat{G}_k^{''}, k = 0,1,\ldots,I$, we can bound the aggregated transition rates for states in $I+1, I+2, \ldots, J$ by the following theorem:

Theorem 4.4 *Given any rate matrix in the form of $\hat{G}_k^{''}, k = 0,1,\ldots,I$, we replace each non-zero transition rate $\hat{q}_{i,j}^{''}, i = I+1,\ldots,J, j = 0,1,\ldots,J$ and $j < i$ by* *a non-zero lower bound and each transition rates* $\hat{q}''_{i,j}, i, j = I+1, \ldots, J$ *and* $j > i$ *by an upper bound. We can obtain lower bound steady state probabilities for states* $i = 0, 1, \ldots, I$

Proof: Based on the result from [MSG89]. □

Since the original rate matrix has a general transition structure, we do not guaranteed that this approach will always provide a tight bounds. But the approach will work well if:

1. the dimension of the northwest corner is large.
2. the density of the northwest corner is sparse.
3. the number of *return states* is relatively small.

4.3 Example

In this section, we illustrate the bounding methodology for a highly fault-tolerant distributed system which is illustrated in Figure 4.3. This distributed system consists of three sub-systems. Each sub-system is composed of a two-processor module, two memory modules, two disk controller modules, and a disk array system [PGK88] consists of three disks with a hot-standby. Due to the fault-tolerant characteristics of disk array, whenever a disk failure occurs, the storage system can still support I/O requests although at a degraded performance mode. We assume we have a hot standby disk in storage system such that we can rebuild the information in the failed disk on the fly [ML90]. Components may fail and be repaired according to the rates given in Table 4.1. If a processor fail, it has a 0.01 probability to write some garbage in the memory module can cause both memory modules to fail. The sub-system is considered failed when:

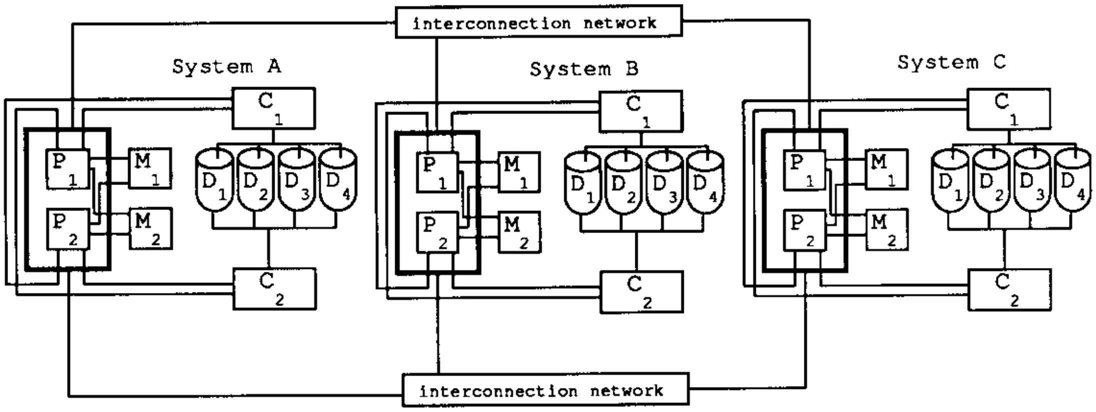

Figure 4.3: A highly fault-tolerant distributed system with module repair.

1. a processor module within a sub-system failed.
2. both disk controllers within a sub-system failed.
3. both memory modules within a sub-system failed.
4. two or more disks within a sub-system failed and,

This highly fault-tolerant distributed system is considered failed when:

1. two of the sub-systems failed.
2. both interconnection networks failed.

Modules are repaired by a single repair service personnel which gives preemptive priority to components in the order:

- processor module
- disk controller
- disk
- memory module

- interconnection network

Since we have a two-processor module, we only replace the module whenever both processors within the same module fail. Also, when the number of failed components in the system exceeds 10, the system manager assume the distributed system is performing poorly in the degraded mode and he will call in an additional repair service personnel. We assume the distributed system is in *active breakdown mode*, meaning that components fail even when the system is non-operational.

Components	Mean Failure Rate	Mean Repair Rate
Processor	1/1000	2.5
Memory	1/1750	2.3
Disk Controller	1/1600	2.2
Disk	1/15000	2.2
Interconnection Network	1/6000	2.4

Table 4.1: Failure and repair rates(per hour).

In Table 4.2, we present the bounds when the transitions are generated in detail for states up to k failures. As illustrated, the tightness of the bound is the function of the details states we want to solve.

k failures	Availability Upper Bound	Availability Lower Bound
0	1.0	0.99767591
1	1.0	0.99923523
2	0.99999998	0.99994371
3	0.99999932	0.99999211
4	0.99999878	0.99999643

Table 4.2: Availability bounds on highly fault-tolerant distributed system.

CHAPTER 5

ALGORITHMIC APPROACH TO BOUNDING THE MEAN RESPONSE TIME OF A MINIMUM EXPECTED DELAY ROUTING

In this chapter we present an algorithmic approach to bounding the mean response time of a multi-server system in which the minimum expected delay routing policy is used, i.e. an arriving job will join the queue which has the minimal expected value of unfinished work. We assume the queueing system to have K servers, each with an infinite capacity queue. The arrival process is Poisson with parameter λ and the service time distribution on server i is exponentially distributed with mean $1/\mu_i, 1 \leq i \leq K$. Without loss of generality, we assume $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$. The computation algorithm allows one to tradeoff accuracy and computational cost. and expected number of customers are computed and the spread between the bounds can be reduced with additional space and time complexity. Examples are presented which illustrate the excellent relative accuracy attainable with relatively little computation.

5.1 Introduction

In this chapter, we are concerned with bounding the mean response time (and thereby the mean number of customers in the system) of the *minimum expected delay* routing policy (a natural generalization of the shortest queue routing policy). The system under study has K servers, where $K \geq 2$. Each server has an infinite capacity queue and service rates are exponentially distributed with rate μ_i, $i = 1, 2, \ldots, K$. Without loss of generality, we assume $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$. The job arrival process is Poisson with rate λ. Upon arrival, the job will join the queue with minimal expected unfinished work (the formal definition of the routing discipline will be given later). In case of a tie, the job will join the server with lowest index. If all the service rates are the same, then the scheduling policy reduces to the classic shortest queue routing policy.

Joining the shortest queue is a natural way to balance the load in a multi-server system and thereby achieve better system performance, i.e. mean response time. One of the major difficulties in analyzing this kind of routing discipline is the multidimensional nature of the state space, which is infinite in each of the K dimensions and the lack of closed form solution. Most of the published results are limited to the case where $K = 2$ with exponential interarrival and service times.

We start with a brief review of the published literature on the shortest queue routing problem. Kingman [Kin61], and later Flatto and McKean [FM77] studied this problem with $K = 2$ via transform methods. They obtained an expression for the mean number of jobs in the system expressed as an infinite sum which can be simplified under a heavy traffic assumption. Cohen and Boxma [CB83] treated a similar problem as a Reimann-Hilbert boundary problem and obtained a functional representation for the mean number of customers in the system. Conolly [Con84] studied the same model as in [FM77, Kin61] and proposed an approximation algorithm for evaluating equilibrium state probabilities via state truncation. Rao and Posner [RP87] proposed an approximation algorithm to analyze a system with $K = 2$ and in which each server has a different service rate (heterogeneous servers). An arriving job joins the server with smaller number of jobs (rather than joining the server with minimum expected delay). The analysis approach involves treating one of the queues has a bounded capacity so that the transition rate matrix for the modified system can be expressed in a matrix-geometric form [Neu81]. Grassman [Gra80] studied the same problem with $K = 2$ and solved for transient and steady state behavior. Halfin [Hal85] studied the two servers problem and used a linear programming technique to compute bounds on the mean number of customers in the system. Blanc [Bla87] studied the join shortest queue problem with arbitrary number of heterogeneous servers. He proposed an approximation method which is based on power series expansions and recursion which requires substantial computational effort. Nelson and Philips [NP89, NP90] proposed an approximation for mean response time with K homogeneous servers. More importantly, the approximation allows general interarrival and service time distribution. Avritzer [Avr90] studied a dynamic load balancing algorithm which used threshold policy in an asymmetric distributed system. The result is only applicable to two distinct servers and a small class of threshold sizes, no formal proof is given on how to obtain performance bounds. None of the work cited above treats more than two servers and simultaneously provides error bounds.

The major contribution of this chapter is a computation algorithm that (1) allows more than $K \geq 2$ servers, (2) allows heterogeneous servers, (3) includes scheduling based on queue length and service rate (thus, a generalization of joining the shortest queue) and (4) provides error bounds. The bounding methodology also allows one to tradeoff accuracy and computational cost as will be demonstrated.

In Section 5.2, we define formally the queueing system we are analyzing. In Section 5.3 and 5.4, we present Markov models which provide upper and lower bounds on the mean response time and formally prove that these modified models do provide bounds. In Section 5.5, we show how we can further reduce the state space by lumping similar states. In Section 5.6, we present two numerical examples and show that the bounds are indeed tight. Conclusions are presented in Section 5.7.

5.2 Minimum Expected Delay Routing Model

We consider a system with $K \geq 2$ servers, each with its own infinite capacity queue and exponential service rate μ_i, $i = 1, 2, \ldots, K$ and $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$. The job arrival process is Poisson with rate λ. Let $n_i(t)$ be the number of customers in the i^{th} server queue at time t. Let $U_i(t) = (1 + n_i(t))/\mu_i$, which is the expected unfinished work at the i^{th} server if the new customer joins queue i. Define $U^*(t) = \min\{U_i(t), i = 1, \ldots, K\}$. Upon arrival of a job at time t, the job will join a server j where $U_j(t) = U^*(t)$. If a tie occurs, the job will join the lowest index server in the set $\{j | U_j(t) = U^*(t)\}$. A special case of this routing discipline is when all service rates are equal, and in this case it reduces to the classic shortest queue routing problem. We can construct a Markov model, M, for this queueing system with state space:

$$\{s \;=\; [n_1, n_2, \ldots, n_K] \,|\, n_i \geq 0, \;\; i = 1, \ldots, K\}$$

Assume the system is stable; that is $\rho = \lambda / \sum_{i=1}^{K} \mu_i < 1$. The steady state probability vector for this continuous-time Markov model is the solution to:

$$\vec{\pi} G \;=\; 0 \tag{5.1}$$

$$\vec{\pi}\underline{e} \;=\; 1 \tag{5.2}$$

where $\vec{\pi}$ is the steady state probability vector, G is the transition rate matrix and $\underline{e}$ denotes an appropriately dimensioned column vector of $1's$.

We can transform this continuous-time Markov model into a discrete-time Markov model via uniformization [Ros83] (the rationale behind this transformation is to facilitate the comparison of the original model and the modified model). To express the one-step transition probabilities for this discrete-time Markov chain, we need the following notation:

$\mathcal{S}$ = total state space of the original model, M.

h = $[\lambda + \sum_{i=1}^{K} \mu_i]^{-1}$

$U^*(s)$ = $\min\{U_i | U_i = (1+n_i)/\mu_i, i = 1,..,K, s = [n_1, n_2, \ldots, n_K]\}$.

$n_a(s)$ = set of servers in state s in which $\{U_i = U^*(s)\}$.

$n^*(s)$ = the lowest index for servers in set $n_a(s)$.

$1\{c\}$ = indicator function for condition c.

The one-step transition probabilities for a given state $s = [n_1, \ldots, n_i, \ldots, n_k]$ are:

$$s \rightarrow \quad s + e_i \qquad 1\{i = n^*(s)\}h\lambda \tag{5.3}$$

$$s \rightarrow \quad s - e_i \qquad 1\{n_i > 0\}h\mu_i \tag{5.4}$$

$$s \rightarrow \quad s \qquad 1 - h[\lambda + \sum_{i=1}^{K} 1\{n_i > 0\}\mu_i] \tag{5.5}$$

where $s + e_i$ is the state s with one additional customer in the i^{th} queue. Also note that:

$$1 - h[\lambda + \sum_{i=1}^{K} 1\{n_i > 0\}\mu_i] \;=\; \sum_{i=1}^{K} 1\{n_i = 0\}\mu_i h \tag{5.6}$$

Let P be the transition probability matrix for the transformed discrete-time Markov chain, we can obtain the steady state probability, at least theoretically, by solving the following system of linear equations:

$$\begin{aligned} \vec{\pi} P &= \vec{\pi} \\ \vec{\pi} \underline{e} &= 1 \end{aligned} \qquad (5.7)$$

Of course, based on the state description, $\vec{\pi}$ is a K-dimensional vector which is infinite in each dimension. The exact solution to this problem has thus far been found to be intractable.

In general, the original problem does not possess a closed form solution and it is impossible to solve the problem numerically due to its state space cardinality. Since the Markov chain lacks special structure, techniques such as the matrix-geometric methods do not apply in general. One natural way to approach this problem is to *construct* another model that closely bounds the performance of the original problem and at the same time, the modified model should have either a known closed form solution or at least be efficiently evaluable by numerical methods.

An important observation is that the motivation for using minimum expected delay policy is to balance the workload among all servers in the systems. Consider a system of two servers with equal service rate in which the current state is $[5, 1]$. The purpose of using the routing policy is to balance the system as much as possible, therefore it is reasonable to assume that a highly unbalanced state(e.g., $[5, 1]$) has a much smaller probability mass than the balanced state (i.e., $[3, 3]$). This crucial insight provides the rational for constructing two modified versions of the original model which can be shown to bound the mean response time of the original system. In both cases we represent the exact behavior (transition rates) for the most "popular" states. The number of states in the most popular subset is a function of the accuracy demanded and computational cost one is willing to pay. When the system leaves this subset we modify the behavior of the system in such a way that (a) the modified system has an efficient solution and (b) the modified model behavior can be shown to bound the behavior of the original model.

In the following two sections, we present two Markov models which can provide an upper bound and a lower bound mean response time. We also present numerical procedures to efficiently solve these two modified models.

5.3 Upper Bound Model

In this section, we construct a modified Markov model, M_u, which provides an upper bound for the mean response time and mean number of customer of the original model, M. For the upper bound model M_u, we assume we have the same system configuration, namely the job arrival process is Poisson with rate λ and system has K servers with service rates $\mu_i, i = 1, 2, \ldots, K$, and $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$.

The upper bound model can be described as follows. There are two additional model parameters for M_u. First, we have a threshold parameter d which indicates the degree of *imbalance* permitted between different servers' queues (a formal definition for d will be given later.) A job may depart from the system only if its departure will not violate the maximum degree of imbalance permitted. If the job departure would violate the threshold setting, the job restarts itself within the same server. Intuitively, this mechanism forces a job to stay in the system at least as long as in the original model and thereby increases the mean number of jobs in the system. The rationale behind the threshold parameter is to generate a model which has a state space which is a small subset of the state space of the original model. The second parameter is the ***artificial capacity***, C_i where $i = 1, 2, ..., K$ (again, C_i will be precisely defined later) for each server. Whenever a job arrives to the system and finds that each server has an integer multiple of C_i jobs, each server will put all jobs in its queue (except the arriving job) into a suspended state and a new busy cycle is started. This busy cycle will end when all servers complete all jobs except for the suspended jobs. The suspended jobs are then released and can be served. Note that the definition here is recursive. During the busy period following suspension of a set of jobs, the capacities C_i can again be exceeded, causing another set of jobs to be suspended. When a busy period ends, only the set of jobs suspended at the initiation of that busy period are released for service. The purpose of the C_i, $1 \leq i \leq K$, is to create a matrix with repetitive structure and based on that structure, we will be able to derive an efficient numerical solution algorithm. The computation algorithm is based on a partitioning of the state space of M_u into $\{\mathcal{S}_0 \cup \mathcal{S}_1 \cdots\}$ such that all states in $\mathcal{S}_i$, $i \geq 0$ satisfy the condition $iC_j \leq n_j \leq (i+1)C_j$ for $j = 1, \ldots, K$. Due to the routing of arrivals and the constraint on departures, we can show that there is only one transition from $\mathcal{S}_i$ to $\mathcal{S}_{i+1}$ and the transitions from $\mathcal{S}_{i+1}$ to $\mathcal{S}_i$ can only go to one state in $\mathcal{S}_i$. This modification to the model should also increase the mean number of jobs in the system compared to the original model since service of a suspended job can only be resumed service when all the active jobs depart from the system.

As an example, assume that we have a system with four homogeneous servers and we let $C_i = 10$, for $i = 1, 2, 3, 4$. It is easy to see that $\mathcal{S}_0$ consists of all states for which each queue has between 0 to 10 customers; $\mathcal{S}_1$ consists of all states for which each queue has 10 suspended customers, and has between 0 to 10 active customers and at least one queue has an active customer. Observe that the only transition from $\mathcal{S}_0$ to $\mathcal{S}_1$ is through state $[10, 10, 10, 10]$. This is due to the routing of arrivals. The only non-zero transitions from $\mathcal{S}_1$ to $\mathcal{S}_0$ are from states $[11, 10, 10, 10]$, $[10, 11, 10, 10]$, $[10, 10, 11, 10]$ and $[10, 10, 10, 11]$ to $[10, 10, 10, 10]$. This is due to the rule introduced in M_u that suspended customers are only served when the busy period (corresponding to states in $\mathcal{S}_1$) has completed. For a heterogeneous server system, the value of C_i has to be chosen to be proportional to the relative service rate in the system to maintain the same structure for the transition rate matrix.

An important point is that the parameters d and C_i can be chosen to control the extent to which M_u behaves like the original model M, i.e. the larger d and C_i are, the larger the portion of the state space that has behavior identical to the original model.

We define the following variables for M_u.

$\mathcal{S}_u$ = total state space of M_u, where $\mathcal{S}_u \subset \mathcal{S}$.

h = $[\lambda + \sum_{i=1}^{K} \mu_i]^{-1}$

C_i = $\lfloor \frac{\mu_i}{\mu_1} C \rfloor$, $i = 1, 2, \ldots, K$, where C is some positive integer such that $\lfloor \frac{\mu_i}{\mu_1} C \rfloor \geq 1$.

d = threshold setting where $(C_1 - C + 1) \leq d \leq C_1$.

$n_{max}(s)$ = $\max\{n_i | s = [n_1, \ldots, n_i, \ldots, n_K]\}$.

$l(s)$ = smallest integer l such that $lC_i - n_i \geq 0$ for all servers i, $i = 0, 1, \ldots, K$ in state s. Note that $l(s)$ is the depth of recursion of job suspensions in state s.

We transform this continuous-time Markov model into a discrete-time Markov chain with the same uniformization parameter h which we used in the original model M. The one-step transition probabilities of the discrete-time Markov chain for a given state $s = [n_1, \ldots, n_i, \ldots, n_k]$ are:

$$s \rightarrow \quad s + e_i \quad 1\{i = n^*(s)\}h\lambda \tag{5.8}$$

$$s \rightarrow \quad s - e_i \quad 1\{n_i > 0\}1\{n_{max}(s) - n_i < d\}1\{n_i - (l(s) - 1)C_i > 0\}h\mu_i \tag{5.9}$$

$$\begin{aligned} s \rightarrow \quad s \quad & 1 - h[\lambda + \sum_{i=1}^{K} 1\{n_i > 0\}1\{n_{max}(s) - n_i < d\} \\ & \qquad 1\{n_i - (l(s) - 1)C_i > 0\}\mu_i] \end{aligned} \tag{5.10}$$

Note that for transition $s \rightarrow s - e_i$, the second indicator function reflects that a job cannot depart if it violates the maximum degree of imbalance permitted. The third indicator function reflects that job cannot depart if it is in a suspended state. We are now in a position to formally compare the original (M) and the modified Markov chain (M_u) and prove that the mean response time of M_u is an upper bound of the mean response time of M.

5.3.1 Proof of upper bound mean response time

Our proof that the mean response time of the modified model is an upper bound on the mean response time of the original model follows the approach in [Dij90]. Let T and T_u be the one-step expectation operators of the original model M and the upper bound Markov model M_u. That is for any non-decreasing function f, we define T in terms of the one-step transition probabilities to be:

$$\begin{aligned} Tf(s) &= \sum_{s' \in S} p[s \rightarrow s']f(s') \\ T_u f(s) &= \sum_{s' \in S_u} p[s \rightarrow s']f(s') \end{aligned}$$

where $p[s \rightarrow s']$ is the transition probability from state s to state s'.

Let R and R_u be the mean response time of M and M_u respectively. And let N and N_u be the mean number of customers in the system for M and M_u. To show $R \leq R_u$, all we need to show is $N \leq N_u$ since the average arrival rate for both models is λ. Define the reward for state s as $r(s) = \sum_{i=1}^{K} n_i$. The mean number of customers in the system can be expressed in term of the expected reward function:

$$N = \sum_{s \in \mathcal{S}} r(s)\pi(s) \tag{5.11}$$

Let $V^t(s)$ be the total expected reward over t periods with one-step reward function r when starting in state s. We have:

$$V^t[s] = \sum_{k=0}^{t} T^k[r(s)]$$

with T^0 being the identity function. By the Markovian property, we have:

$$V^t[s] = r(s) + TV^{t-1}[r(s)]$$

Since both Markov models are irreducible (easily seen from their definitions), steady state performance measures are independent of the initial state s', and we have:

$$N = \sum_{s \in \mathcal{S}} r(s)\pi(s) = \lim_{t \to \infty} \frac{1}{t} V^t[s'] \qquad \textit{and} \tag{5.12}$$

$$N_u = \sum_{s \in \mathcal{S}_u} r(s)\pi_u(s) = \lim_{t \to \infty} \frac{1}{t} V_u^t[s'] \tag{5.13}$$

By comparing the total expected reward over t periods for both the upper bound and original model and $s \in \mathcal{S}_u$, we have:

$$\begin{aligned}(V_u^t - V^t)[s] &= r_u(s) - r(s) + (T_u V_u^{t-1} - TV^{t-1})[s] \\ &= (T_u - T)V^{t-1}[s] + T_u(V_u^{t-1} - V^{t-1})[s] \\ &= \sum_{k=0}^{t-1} T_u^k (T_u - T)V^{t-k-1}[s]\end{aligned}$$

The last expression was obtained based on the recursive definition and $V^0(s) = 0$ for all s. Now divide both sides by t and take the limit as t goes to infinity. We can conclude that $N \leq N_u$ if for all $s \in \mathcal{S}_u$ and for $t \geq 0$:

$$(T_u - T)V^t[s] \;\geq\; 0 \tag{5.14}$$

Based on the definition on the one-step expectation operator on the original model M and the upper bound model M_u, we have the following relationship. That is for any state $s \in \mathcal{S}_u$:

$$\begin{aligned}(T_u - T)f(s) \;=\; & \sum_{i=1}^{K} 1\{n_i > 0\}(1\{n_{max}(s) - n_i = d\} \mid \\ & 1\{n_i - (l(s)-1)C_i = 0\})\mu_i h[f(s) - f(s-e_i)]\end{aligned} \tag{5.15}$$

where symbol "|" is the logical OR operator. Substituting $f(s)$ for $V^t(s)$, it follows easily that Equation (5.14) will be satisfied if the following conditions are satisfied:

$$V^t[s] - V^t[s-e_i] \;\geq\; 0 \qquad for\ i = 1, 2, \ldots, K\ ;\ t \geq 0;\ \ n_i > 0\ and\ s \in \mathcal{S}_u \tag{5.16}$$

Theorem 5.1

$$V^t[s] - V^t[s-e_i] \geq 0 \qquad for\ i = 1, 2, \ldots, K\ ;\ t \geq 0;\ \ n_i > 0\ and\ s \in \mathcal{S}_u$$

Proof: The proof is by induction on t. When $t = 0$, $V^0(s) = 0$ for all s, therefore the condition is satisfied. Now assume the condition is satisfied for $t = m$. For $t = m + 1$, we have in general[1]:

$$V^{m+1}(s) - V^{m+1}(s-e_i) = \Bigg\{ r(s) + \sum_{j=1}^{K} \lambda h 1\{j = n^*(s)\} V^m(s+e_j) +$$

[1] Note that the condition implies that in state s, there is at least one job in the i^{th} queue.

$$\sum_{j=1,j\neq i}^{K} 1\{n_j > 0\}\mu_j h V^m(s-e_j) +$$

$$\mu_i h V^m(s-e_i) + \sum_{j=1,j\neq i}^{K} 1\{n_j=0\}\mu_j h V^m(s)\Bigg\} -$$

$$\Bigg\{ r(s-e_i) + \sum_{j=1}^{K}\lambda h 1\{j = n^*(s-e_i)\}V^m(s-e_i+e_j)+$$

$$\sum_{j=1,j\neq i}^{K} 1\{n_j>0\}\mu_j h V^m(s-e_i-e_j) +$$

$$1\{n_i-1>0\}\mu_i h V^m(s-e_i-e_i) +$$

$$\left(\sum_{j=1,j\neq i}^{K} 1\{n_j=0\}\mu_j h + 1\{n_i-1=0\}\mu_i h\right) V^m(s-e_i)\Bigg\}$$

Grouping similar terms, we have:

$$\begin{aligned}
V^{m+1}(s) - V^{m+1}(s-e_i) \;=\; & \Bigg\{ \Bigg[r(s) - r(s-e_i) \Bigg] + \\
& \Bigg[\sum_{j=1}^{K} \lambda h 1\{j = n^*(s)\} V^m(s+e_j) - \\
& \qquad \sum_{j=1}^{K} \lambda h 1\{j = n^*(s-e_i)\} V^m(s-e_i+e_j) \Bigg] + \\
& \sum_{j=1,j\neq i}^{K} 1\{n_j>0\}\mu_j h \Bigg[V^m(s-e_j) - V^m(s-e_i-e_j) \Bigg] + \\
& \Bigg[\mu_i h V^m(s-e_i) - 1\{n_i-1>0\}\mu_i h V^m(s-e_i-e_i) \\
& \qquad -1\{n_i-1 = 0\}\mu_i h V^m(s-e_i) \Bigg] \\
& \sum_{j=1,j\neq i}^{K} 1\{n_j=0\}\mu_j h \Bigg[V^m(s) - V^m(s-e_i) \Bigg] \Bigg\}
\end{aligned}$$

It is clear that the first [] term is greater than zero. By induction hypothesis, the third, fourth, and fifth [] terms are greater than zero. It remains to prove that the second [] term is greater than or equal to zero. To answer this question, we break this term into four cases.

For case 1, for state s, $i \neq n^*(s)$ and for state $s-e_i$, $i \neq n^*(s-e_i)$, this implies $n^*(s) = n^*(s-e_i) = j$ where $j \neq i$, the second term is:

$$\sum_{j=1,j\neq i}^{K} \lambda h 1\{j = n^*(s)\} \left[(V^m(s+e_j) - V^m(s-e_i+e_j) \right] \quad \geq \quad 0$$

Case 2: for state s, $i = n^*(s)$ and for state $s-e_i$, $i = n^*(s-e_i)$, the second term is:

$$\lambda h \left[(V^m(s+e_i) - V^m(s-e_i+e_i) \right] \quad \geq \quad 0$$

Case 3, for state s, $i \neq n^*(s)$ and for state $s-e_i$, $i = n^*(s-e_i)$, the second term is:

$$\left[\sum_{j=1,j\neq i}^{K} \lambda h 1\{j = n^*(s)\} V^m(s+e_j) - V^m(s- e_i+ e_i) \right] \quad \geq \quad 0$$

Case 4. For state s, $i = n^*(s)$ and for state $s-e_i$, $i \neq n^*(s-e_i)$. This case is obviously impossible. □

5.3.2 Computational algorithm for solving the upper bound model

In this section, we will describe an algorithm for computing the mean response time of the upper bound model. We define a partition of the state space of M_u, $\mathcal{S}_u = \cup_{i=0}^{\infty} \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $\forall i \neq j$, where:

$\mathcal{S}_0$ = set of states with $n_j \leq C_j$, $j = 1, 2, \ldots, K$.

$\mathcal{S}_i$ = set of states with $iC_j \leq n_j \leq (i+1)C_j$, $j = 1, 2, \ldots, K$ and for $i \geq 1$.

$P_{\mathcal{S}_i, \mathcal{S}_j}$ = transition probability matrix from states in $\mathcal{S}_i$ to states in $\mathcal{S}_j$.

The transition rate matrix P_u has the form depicted in Figure 5.1:

$$P_u = \left[\begin{array}{cccccc} P_{\mathcal{S}_0,\mathcal{S}_0} & P_{\mathcal{S}_0,\mathcal{S}_1} & 0 & 0 & 0 & \cdots \\ P_{\mathcal{S}_1,\mathcal{S}_0} & P_{\mathcal{S}_1,\mathcal{S}_1} & P_{\mathcal{S}_1,\mathcal{S}_2} & 0 & 0 & \cdots \\ 0 & P_{\mathcal{S}_2,\mathcal{S}_1} & P_{\mathcal{S}_2,\mathcal{S}_2} & P_{\mathcal{S}_2,\mathcal{S}_3} & 0 & \cdots \\ 0 & 0 & P_{\mathcal{S}_3,\mathcal{S}_2} & P_{\mathcal{S}_3,\mathcal{S}_3} & P_{\mathcal{S}_3,\mathcal{S}_4} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \right.$$

Figure 5.1: Transition probability matrix for upper bound model.

This is a block tridiagonal transition probability matrix and therefore represents a generalized birth-death process. By aggregating each partition $\mathcal{S}_i$, we can form a birth-death process. Next, we show how to obtain the exact conditional state probability vector, given that the system is in partion $\mathcal{S}_i$. Once we have this information, it follows easily that we can obtain the aggregate transition probabilities exactly.

There are several important features of this upper bound model, M_u. First, there is only a single state in $\mathcal{S}_i$ that has a non-zero transition probability into any states in $\mathcal{S}_{i+1}$, $i \geq 0$. Let us call this state $s_i(C_0)$. State $s_i(C_0)$ is:

$$s_i(C_0) = [n_1, n_2, \ldots, n_K] \in \mathcal{S}_i \quad \textit{where} \quad n_j = (i+1)C_j \;\; \forall \;\; j = 1, 2, \ldots, K$$

This follows from the rule used to assign an arriving customer to a server. Also, there are K states from $\mathcal{S}_i$ that have non-zero transition probabilities to states in $\mathcal{S}_{i-1}$ Each corresponds to which server is the last to complete its "active" (non-suspended) customers. where $i \geq 1$. Let us call these states $s_i(l)$, $1 \leq l \leq K$, $i \geq 1$. These states are:

$$s_i(l) \;\; = \;\; [n_1, n_2, \ldots, n_K] \in \mathcal{S}_i \;\; \textit{where}$$

$$\begin{aligned} n_l &= iC_l + 1 \;\; and \\ n_j &= iC_j \;\; for \;\; l \neq j \;\; and \;\; l, j = 1, 2, \ldots, K \end{aligned}$$

This follows from the restrictions on departures in the upper bound model. The following are easily seen to be the transition probabilities between $s_i(C_0)$ and $s_{i+1}(l), l = 1, 2, \ldots, K$ are:

$$\begin{aligned} s_i(C_0) &\rightarrow s_{i+1}(l) & & 1\{l = n^*(s_i(C_0))\}\lambda h \\ s_{i+1}(l) &\rightarrow s_i(C_0) & & \mu_l h \;\; for \;\; l = 1, 2, \ldots, K \end{aligned}$$

Another important observation is that the submatrices $P_{\mathcal{S}_i,\mathcal{S}_i}$ for $i \geq 1$ are all identical. We now consider how to compute the conditional state probabilities $P\{s \in \mathcal{S}_i | \mathcal{S}_i\}$ exactly. We first need the following result from [CS86]:

Theorem 5.2 *Given a irreducible Markov process with state space $S = \{A \cup B\}$ and transition probability matrix:*

$$\begin{bmatrix} P_{A,A} & P_{A,B} \\ P_{B,A} & P_{B,B} \end{bmatrix}$$

where $P_{i,j}$ is the transition probability sub-matrix from partition i to j. If $P_{B,A}$ has all zero entries except for some non-zero entries in the i^{th} column, the conditional steady state probability vector given that the system is in partition A is the solution for the following linear system of equations:

$$\begin{aligned} \vec{\pi}_{|A} \left[P_{A,A} + P_{A,B} \; \underline{e} \; \underline{e}_i^T \right] &= \vec{\pi}_{|A} \\ \vec{\pi}_{|A} \; \underline{e} &= 1 \end{aligned}$$

where $\underline{e}_i^T$ is a row vector 0 in each component except the i^{th} component which has the value 1.

We are now in the position to compute the conditional state probabilities on each partition exactly. Without loss of generality, let us consider $\mathcal{S}_i$, for some $i \geq 1$.

Lemma 5.1 *Let $\tilde{P}_{\mathcal{S}_i,\mathcal{S}_i}$ be the transition probability matrix which is similar to $P_{\mathcal{S}_i,\mathcal{S}_i}$ except for the following modification:*

$$\begin{aligned}
\tilde{p}_{s_i(C_0),s_i(C_0)} &= p_{s_i(C_0),s_i(C_0)} + \lambda h && && (5.17)\\
\tilde{p}_{s_i(l),s_i(l)} &= p_{s_i(l),s_i(l)} + \mu_l h && \textit{where } l = n^*(s_{i-1}(C_0)) && (5.18)\\
\tilde{p}_{s_i(j),s_i(l)} &= \mu_i h && j = 1, 2, \ldots, K \textit{ and } j \neq l && (5.19)
\end{aligned}$$

The solution for the following linear system of equations:

$$\begin{aligned}
\vec{\tilde{\pi}} \tilde{P}_{\mathcal{S}_i,\mathcal{S}_i} &= \vec{\tilde{\pi}}\\
\vec{\tilde{\pi}} \underline{e} &= 1
\end{aligned}$$

provides the conditional steady state probability of state s given the Markov chain is in some state in $\mathcal{S}_i$, that is:

$$\tilde{\pi}(s) = \frac{\pi(s)}{\sum_{s \in \mathcal{S}_i} \pi(s)} \qquad \forall\ s \in \mathcal{S}_i$$

Proof: Let us partition the state space $\mathcal{S}_u = \{\mathcal{S}_i^{'} \cup \mathcal{S}_i^{''}\}$ where $\mathcal{S}_i^{'} = \cup_{j=0}^{j=i} \mathcal{S}_j$ and $\mathcal{S}_i^{''} = \mathcal{S}_u - \mathcal{S}_i^{'}$. There is only a single return state in $\mathcal{S}_i^{'}$, which is $s_i(C_0)$, from states in $\mathcal{S}_i^{''}$. Based on theorem 5.2, the modification according to Equation (5.17) provides the conditional steady state probability given the system is in $\mathcal{S}_i^{'}$. Now partition the state space $\mathcal{S}_i^{'} = \{\mathcal{S}_i^1 \cup \mathcal{S}_i\}$ where $\mathcal{S}_i^1 = \cup_{j=0}^{i-1} \mathcal{S}_j$. Note that there is only one return state in $\mathcal{S}_i$, which is $s_i(n^*(s_{i-1}(C_0)))$. Again, based on theorem

5.2, the modification according to equation (5.18) and equation (5.19) provides the conditional state probability vector given the system is in state $\mathcal{S}_i$. □

Since we can compute the conditional state probabilities for each partition $\mathcal{S}_i$ exactly, we can exactly aggregate states in each $\mathcal{S}_i$ into a single state $s_i, i \geq 0$. The aggregate chain is depicted in Figure 5.2.

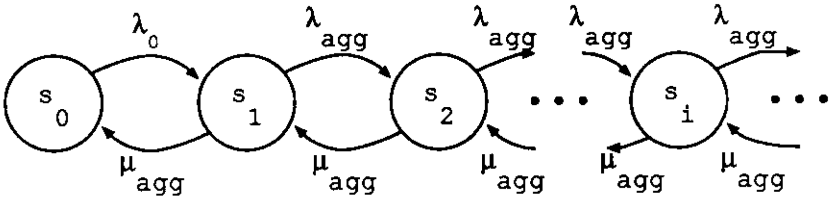

Figure 5.2: Aggregate Chain for upper bound model

$$\begin{aligned}\lambda_0 &= \tilde{\pi}(s_0(C_0))\ \lambda\ h \\ \lambda_{agg} &= \tilde{\pi}(s_i(C_0))\ \lambda\ h \\ \mu_{agg} &= \sum_{l=1}^{K} \tilde{\pi}(s_i(l))\ \mu_l\ h\end{aligned}$$

Solving this chain, we have:

$$\pi(s_0) = \left[1 + \frac{\lambda_0}{\mu_{agg} - \lambda_{agg}}\right]^{-1} \tag{5.20}$$

$$\pi(s_i) = \left[1 + \frac{\lambda_0}{\mu_{agg} - \lambda_{agg}}\right]^{-1} \left(\frac{\lambda_0}{\mu_{agg}}\right) \left(\frac{\lambda_{agg}}{\mu_{agg}}\right)^{i-1} \qquad for \quad i = 1, 2, \ldots \tag{5.21}$$

To obtain the mean number of customers in the the upper bound model, N_u, let us define the following:

$$\begin{aligned}C_0 &= \sum_{i=1}^{K} C_i \\ \tilde{r}(s) &= r(s) - iC_0 \qquad s \in \mathcal{S}_i \\ \tilde{N}(s_i) &= \sum_{s \in \mathcal{S}_i} \tilde{r}(s)\tilde{\pi}(s)\end{aligned}$$

where $\tilde{\pi}(s)$ is the solution of the following Markov chain:

$$\begin{aligned}\vec{\tilde{\pi}}\tilde{P}_{S_i,S_i} &= \vec{\tilde{\pi}} \\ \vec{\tilde{\pi}}\underline{e} &= 1\end{aligned}$$

Then we have:

$$N_u = \tilde{N}(s_0)\pi(s_0) + \sum_{i=1}^{\infty}\left[\tilde{N}(s_i) + iC_0\right]\pi(s_i) \tag{5.22}$$

Since $N(s_i) = N(s_j)$ for $i \neq j$ and $i, j \geq 1$, we can simplify the expression above and obtain the expression for N_u:

$$N_u = \tilde{N}(s_0)\pi(s_0) + \tilde{N}(s_i)(1 - \pi(s_0)) + C_0\lambda_0\frac{\mu_{agg}}{(\mu_{agg} - \lambda_{agg})^2}\pi(s_0) \tag{5.23}$$

From Little's Result [Lit67], the upper bound mean system response time is:

$$R_u = N_u/\lambda \tag{5.24}$$

5.4 Lower Bound Model

In this section, we construct a modified Markov model, M_l, which provides a lower bound for the mean response time of the original model, M. We first give an informal description and motivation for the lower bound model. As for the upper bound model, two additional parameters are used to specify model M_l, namely, d and C_i, $i = 1, \ldots, K$. A job may depart normally from the system only if the departure does not violate the maximum degree of imbalance permitted. If a job departure violates this threshold setting, the system will go into a *full service mode.* In this mode, the system behaves like an $M/M/K$ system with a special service discipline; specifically, if there are j customers (where $j \leq K$)

in the system, these j customers will be executing on the j fastest servers. If there are more than K customers in the system, then the system behaves like a regular $M/M/K$ system. The system will operate in this mode until the next idle time and then it will start behaving like the original system again. Also, when a job arrives and finds that the system has C_f customers, where $C_f = \sum_{i=1}^{K} C_i$, the system will again operate in a full service mode until the system goes idle and then it reverts back to its original behavior. Intuitively, these modifications will yield a lower bound on mean response time. Since the modification are idealization in which either the model behaves exactly as the original model or the best possible service rate is delivered. While this is intuitive we will also formally prove that the modified model M_l yields a lower bound on the mean response time. Of course, it is intended that d and $C_i, i = 1, 2, \ldots, K$ be chosen large enough so that most of the time, M_l behaves like the original model. On the other hand, to be able to solve the model efficiently, we would like to keep these parameters small.

5.4.1 Proof of lower bound mean response time

In order to facilitate the comparison between M and M_l, we organize the state space for model M using the following notation:

$\mathcal{N}_i$ = set of states with exactly i jobs in the system, where $i = 0, 1, \ldots$

$\mathcal{G}$ = $\{\mathcal{N}_1 \cup \mathcal{N}_2 \cup \cdots \mathcal{N}_{C_f}\}$

$Q_{i,j}$ = submatrix containing transition rates from states in $\mathcal{N}_i$ to states in $\mathcal{N}_j$

$Q_{i,\mathcal{G}}$ = submatrix containing transition rates from states in $\mathcal{N}_i$ to states in $\mathcal{G}$

Figure 5.3 illustrates the form of the transition rate matrix for model M when states are ordered according to the number of customers in the system.

$$\left[\begin{array}{cc||cccc}
Q_{0,0} & Q_{0,\mathcal{G}} & 0 & 0 & 0 & \cdots \\
Q_{\mathcal{G},0} & Q_{\mathcal{G},\mathcal{G}} & Q_{\mathcal{G},C_f+1} & 0 & 0 & \cdots \\
\hline\hline
0 & Q_{C_f+1,\mathcal{G}} & Q_{C_f+1,C_f+1} & Q_{C_f+1,C_f+2} & 0 & \cdots \\
0 & 0 & Q_{C_f+2,C_f+1} & Q_{C_f+2,C_f+2} & Q_{C_f+2,C_f+3} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right.$$

Figure 5.3: Transition rate matrix for M.

Using the state replication technique from [MSG89], it is easy to show that we can transform the model M into another model, M_1, by duplicating states in $\mathcal{G}$ without perturbing the expected number of customer in the system. Let us call the duplicated set of states $\mathcal{G}'$. The transition rate matrix M_1, which results from the duplication of states in $\mathcal{G}$, is illustrated in Figure 5.4. More Formally, if[2]:

$$[\underline{\pi}_0, \underline{\pi}_{\mathcal{G}}, \underline{\pi}_{>\mathcal{G}}]$$

is the steady state solution for model M, the steady state probability vector for model M_1 is:

$$\left[\underline{\pi}_0, \underline{\pi}'_{\mathcal{G}}, \underline{\pi}'_{\mathcal{G}'}\underline{\pi}_{>\mathcal{G}}\right] \qquad\qquad where \;\; \underline{\pi}_{\mathcal{G}} = \underline{\pi}'_{\mathcal{G}} + \underline{\pi}'_{\mathcal{G}'}$$

Note that there is a one to one mapping between states in $\mathcal{G}$ and states in $\mathcal{G}'$ and $Q_{\mathcal{G}',\mathcal{G}'} = Q_{\mathcal{G},\mathcal{G}}$, $Q_{\mathcal{G},0} = Q_{\mathcal{G}',0}$ and $Q_{C_f+1,\mathcal{G}} = Q_{C_f+1,\mathcal{G}'}$. Starting from an

[2] to simply notation, we use $\underline{\pi}_{>\mathcal{G}}$ to represent steady state probabilities for states other than state 0 and states in $\mathcal{G}$

$$\left[\begin{array}{cc||ccccc}
Q_{0,0} & Q_{0,\mathcal{G}} & 0 & 0 & 0 & 0 & \cdots \\
Q_{\mathcal{G},0} & Q_{\mathcal{G},\mathcal{G}} & 0 & Q_{\mathcal{G},C_f+1} & 0 & 0 & \cdots \\
\hline\hline
Q_{\mathcal{G}',0} & 0 & Q_{\mathcal{G}',\mathcal{G}'} & Q_{\mathcal{G}',C_f+1} & 0 & 0 & \cdots \\
0 & 0 & Q_{C_f+1,\mathcal{G}'} & Q_{C_f+1,C_f+1} & Q_{C_f+1,C_f+2} & 0 & \cdots \\
0 & 0 & 0 & Q_{C_f+2,C_f+1} & Q_{C_f+2,C_f+2} & Q_{C_f+2,C_f+3} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right.$$

Figure 5.4: Transition rate matrix for M_1.

empty system, only states in $\mathcal{G}$ are visited until the number in system exceeds C_f. When the number in system falls to C_f again, states in $\mathcal{G}'$ will be visited rather than states in $\mathcal{G}$ until the system goes idle. At this point the described behavior repeats. Intuitively, the idea is that if C_f is large enough, the number in system only rarely exceeds C_f and therefore most of the time, M_l behaves exactly as the original model M.

Although the states in $\mathcal{G}$ are more popular that other states in the model, there are still a large number of states in $\mathcal{G}$ which have low steady state probability, for example, those states with large imbalance in queue length. With this in mind, let us partition $\mathcal{G}$ into two sets of states, $\mathcal{G}_1$ and $\mathcal{G}_2$ where $\mathcal{G}_1$ contains all those states that satisfy the threshold setting d, and $\mathcal{G}_2 = \mathcal{G} - \mathcal{G}_1$. Based on the results from [LM90], transitions from $\mathcal{G}_1$ to $\mathcal{G}_2$ can be transformed to transitions from $\mathcal{G}_1$ to the corresponding states in $\mathcal{G}'$ (since there is a one to one mapping between states in $\mathcal{G}$ and $\mathcal{G}'$) without perturbing the mean number of customers in the systems. Formally, the steady state probability vector for model M_2 is:

$$\left[\underline{\pi}_0, \underline{\pi}''_{\mathcal{G}_1}, \underline{\pi}''_{\mathcal{G}'}\underline{\pi}_{>\mathcal{G}}\right] \qquad \textit{where} \;\; \underline{\pi}_{\mathcal{G}} = \left[\underline{\pi}''_{\mathcal{G}_1}, \underline{0}\right] + \underline{\pi}''_{\mathcal{G}'} = \underline{\pi}'_{\mathcal{G}} + \underline{\pi}'_{\mathcal{G}'}$$

The transition rate matrix for this new model M_2 is illustrated in Figure 5.5.

$$\left[\begin{array}{cc||ccccc}
Q_{0,0} & Q_{0,\mathcal{G}_1} & 0 & 0 & 0 & 0 & \cdots \\
Q_{\mathcal{G}_1,0} & Q_{\mathcal{G}_1,\mathcal{G}_1} & Q_{\mathcal{G}_1,\mathcal{G}'} & Q_{\mathcal{G}_1,C_f+1} & 0 & 0 & \cdots \\
\hline\hline
Q_{\mathcal{G}',0} & 0 & Q_{\mathcal{G}',\mathcal{G}'} & Q_{\mathcal{G}',C_f+1} & 0 & 0 & \cdots \\
0 & 0 & Q_{C_f+1,\mathcal{G}'} & Q_{C_f+1,C_f+1} & Q_{C_f+1,C_f+2} & 0 & \cdots \\
0 & 0 & 0 & Q_{C_f+2,C_f+1} & Q_{C_f+2,C_f+2} & Q_{C_f+2,C_f+3} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right.$$

Figure 5.5: Transition rate matrix for M_2.

Now, (conceptually) we apply exact aggregation [Cou77] to states in $\mathcal{G}'$ and to states in $\mathcal{N}_i$ for $i > C_f$. That is, we aggregate all states with equal number of customer into a single state. Denote the aggregate state corresponding to i customers in the systems as a_i and let $g_{i,j}$ be the aggregate rate between aggregate state i and j. The transition rate matrix for this model M_3, is illustrated in Figure 5.6.

We are now in a position to compare model M_3 (which has the same expected mean number of customers as the original model, M) to the lower bound model M_l since they have similar transition structure. Note that in the lower bound model M_l, the system operates in the *full service mode* when it is in states a_i, $i \geq 1$. That is[3]:

$$g^*_{a_i,a_{i-1}} = \begin{cases} \sum_{j=1}^{i} \mu_j & 1 \leq i \leq K \\ \sum_{j=1}^{K} \mu_j & i > K \end{cases} \tag{5.25}$$

[3] To simplify notation, we use notation a_0 (a state with no customers in the system) and 0 interchangeably

$$\left[\begin{array}{cc||ccccc}
Q_{0,0} & Q_{0,\mathcal{G}_1} & 0 & 0 & 0 & 0 & \cdots \\
Q_{\mathcal{G}_1,0} & Q_{\mathcal{G}_1,\mathcal{G}_1} & Q_{\mathcal{G}_1,a_1} & Q_{\mathcal{G}_1,a_2} & Q_{\mathcal{G}_1,a_3} & Q_{\mathcal{G}_1,a_4} & \cdots \\
\hline\hline
g_{a_1,0} & 0 & g_{a_1,a_1} & g_{a_1,a_2} & 0 & 0 & \cdots \\
0 & 0 & g_{a_2,a_1} & g_{a_2,a_2} & g_{a_2,a_3} & 0 & \cdots \\
0 & 0 & 0 & g_{a_3,a_2} & g_{a_3,a_3} & g_{a_3,a_4} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right.$$

Figure 5.6: Transition rate matrix for M_3.

It is clear that these aggregate rates $g^*_{a_i,a_{i-1}}$ in M_l are upper bounds for aggregate transition rates $g_{a_i,a_{i-1}}$ in M_3.

Again, to facilitate a formal proof that M_l provides a lower bound, we transform the two continuous time Markov modes, M_3 and M_l, into discrete-time Markov chains with the uniformization parameter h. We can then apply the same approach as in Section 5.3.1 to show that the expected number of customers in the system for model M_l is less than the expected number of customers in model M_3. Based on the difference of the one-step expectation operator T_l (for model M_l) and T, we need the following conditions to hold:

$$V^t(a_{i-1}) - V^t(a_i) \ \leq \ 0 \qquad i \geq 1 \ and \ t \geq 0 \tag{5.26}$$

Theorem 5.3

$$V^t(a_{i-1}) - V^t(a_i) \leq 0 \qquad i \geq 1 \ and \ t \geq 0$$

Proof: Let us pick any i, where $i \geq 1$. Again, the proof is by induction. When $t = 0$, the condition is clearly satisfied. Assume the condition holds for $t = m$.

For $t = m+1$ we have in general:

$$\begin{aligned} V^{m+1}(a_{i-1}) - V^{m+1}(a_i) &= \Big\{ r(a_{i-1}) + \lambda h V^m(a_i) + g_{a_{i-1},a_{i-2}} h V^m(a_{i-2}) + \\ &\quad + \Big(1 - h(\lambda + g_{a_{i-1},a_{i-2}})\Big) V^m(a_{i-1}) \Big\} - \\ &\quad \Big\{ r(a_i) + \lambda h V^m(a_{i+1}) + g_{a_i,a_{i-1}} h V^m(a_{i-1}) + \\ &\quad + \Big(1 - h(\lambda + g_{a_i,a_{i-1}})\Big) V^m(a_i) \Big\} \end{aligned}$$

Since the following inequalities hold:

$$\begin{aligned} g_{a_{i-1},a_{i-2}} &\leq g_{a_i,a_{i-1}} \\ \Big[1 - h(\lambda + g_{a_{i-1},a_{i-2}})\Big] &\geq \Big[1 - h(\lambda + g_{a_i,a_{i-1}})\Big] \\ h(g_{a_i,a_{i-1}} - g_{a_{i-1},a_{i-2}}) &= \Big[1 - h(\lambda + g_{a_{i-1},a_{i-2}})\Big] - \Big[1 - h(\lambda + g_{a_i,a_{i-1}})\Big] \end{aligned}$$

By rearranging terms, we have:

$$\begin{aligned} V^{m+1}(a_{i-1}) - V^{m+1}(a_i) &= [r(a_{i-1}) - r(a_i)] + \\ &\quad \lambda h [V^m(a_i) - V^m(a_{i+1})] + \\ &\quad (g_{a_{i-1},a_{i-2}}) h [V^m(a_{i-2}) - V^m(a_{i-1})] + \\ &\quad (g_{a_i,a_{i-1}} - g_{a_{i-1},a_{i-2}}) h [V^m(a_{i-1}) - V^m(a_{i-1})] + \\ &\quad (1 - h(\lambda + g_{a_i,a_{i-1}})) [V^m(a_{i-1}) - V^m(a_i)] \end{aligned}$$

The first [] term is less than zero, the fourth [] term is equal to zero. By the induction hypothesis, the second, third and fifth [] terms are less than or equal to zero. □

5.4.2 Computational algorithm for solving the lower bound model

In this section, we describe an algorithm for computing the mean response time in the lower bound model M_l. Let $\mathcal{S}_0 = \{n_0 \cup \mathcal{G}_1\}$. Again, the transition rate matrix is depicted in Figure 5.7:

$$
\left[\begin{array}{cc||ccccc}
Q_{0,0} & Q_{0,\mathcal{G}_1} & 0 & 0 & 0 & 0 & \cdots \\
Q_{\mathcal{G}_1,0} & Q_{\mathcal{G}_1,\mathcal{G}_1} & Q_{\mathcal{G}_1,a_1} & Q_{\mathcal{G}_1,a_2} & Q_{\mathcal{G}_1,a_3} & Q_{\mathcal{G}_1,a_4} & \cdots \\
\hline\hline
g^*_{a_1,0} & 0 & g^*_{a_1,a_1} & g^*_{a_1,a_2} & 0 & 0 & \cdots \\
0 & 0 & g^*_{a_2,a_1} & g^*_{a_2,a_2} & g^*_{a_2,a_3} & 0 & \cdots \\
0 & 0 & 0 & g^*_{a_3,a_2} & g^*_{a_3,a_3} & g^*_{a_3,a_4} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right.
$$

Figure 5.7: Transition rate matrix for lower bound model.

Observe that if we know the conditional state probabilities for states in $\mathcal{S}_0$, we can aggregate $\mathcal{S}_0$ as a single state, s_0, and we will have an efficient algorithm to compute the mean number of customer in the system. Based on Theorem 5.2, Noted that there is only a single return state to $\mathcal{S}_0$ from states outside $\mathcal{S}_0$, and based on Theorem 5.2, the state probabilities conditioned on the system being in $\mathcal{S}_0$ can be obtained by solving the following system of linear equations:

$$
\begin{aligned}
\vec{\pi}(\mathcal{S}_0)\left[Q_{\mathcal{S}_0,\mathcal{S}_0} + \left(\sum_{i=1}^{C_f+1} Q_{\mathcal{S}_0,\mathcal{N}_i}\,\underline{e}\right)\underline{e}_0^T\right] &= 0 \\
\vec{\pi}(\mathcal{S}_0)\,\underline{e} &= 1
\end{aligned}
$$

where $\vec{\pi}(\mathcal{S}_0)$ is the steady state probability vector given the system is in $\mathcal{S}_0$. We can now apply exact aggregation and the aggregated process is depicted in Figure 5.8.

The transition rates for the aggregated chain are:

$$
g^*_{s_0,a_i} = \vec{\pi}(\mathcal{S}_0)Q_{\mathcal{S}_0,\mathcal{N}_i}\underline{e} \qquad\qquad i = 1,\ldots,C_f+1
$$

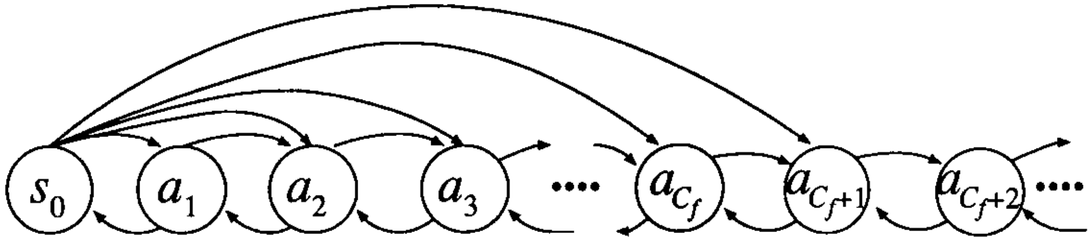

Figure 5.8: Aggregate Chain for lower bound model

$$
\begin{aligned}
g^*_{a_i,a_{i+1}} &= \lambda \qquad\qquad i \geq 1 \\
g^*_{a_1,s_0} &= \mu_1 \\
g^*_{a_i,a_{i-1}} &= \begin{cases} \sum_{j=1}^{i} \mu_j & i = 2, 3, \ldots, K \\ \mu^* & \text{otherwise} \end{cases}
\end{aligned}
$$

where $\mu^* = \sum_{i=1}^{K} \mu_i$.

Solving the chain, we have:

$$
\pi(s_0) = \left[1 + \sum_{i=1}^{C_f+1} \sum_{j=1}^{i} [\lambda^{i-j} (\sum_{k=j}^{C_f+1} g^*_{s_0,a_j}) (\prod_{k=j}^{i} g^*_{a_k,a_{k-1}})^{-1}] + \frac{\lambda}{\mu^* - \lambda} \sum_{j=1}^{C_f+1} [\lambda^{C_f+1-j} (\sum_{k=j}^{C_f+1} g^*_{s_0,a_j}) (\prod_{k=j}^{C_f+1} g^*_{a_k,a_{k-1}})^{-1}] \right]^{-1} \tag{5.27}
$$

$$
\pi(a_i) = \pi(s_0) \sum_{j=1}^{i} [\lambda^{i-j} (\sum_{k=j}^{C_f+1} g^*_{s_0,a_j}) (\prod_{k=j}^{i} g^*_{a_k,a_{k-1}})^{-1}] \qquad i = 1, \ldots, C_f+1 \tag{5.28}
$$

$$
\pi(a_i) = \pi(s_0) (\frac{\lambda}{\mu^*})^{i-C_f-1} \sum_{j=1}^{C_f+1} [\lambda^{C_f+1-j} (\sum_{k=j}^{C_f+1} g^*_{s_0,a_j}) (\prod_{k=j}^{C_f+1} g^*_{a_k,a_{k-1}})^{-1}] \qquad i = C_f+2, \ldots \tag{5.29}
$$

To obtain the mean number of customers in the system, N_l and the mean response time R_l, let

$$
\tilde{N}(S_0) = \sum_{s \in S_0} r(s) \tilde{\pi}(s)
$$

Then we have:

$$
\begin{aligned}
N_l &= \tilde{N}(s0)\pi(\mathcal{S}_0) + \sum_{i=1}^{\infty} i\pi(a_i) && (5.30)\\
R_l &= N_l/\lambda && (5.31)
\end{aligned}
$$

5.5 Further State Space Reduction

In the previous sections, we discussed the methodology to construct an upper bound model M_u and a lower bound model M_l. The computational costs in solving the models are:

1. obtaining the conditional state probabilities in $\mathcal{S}_0$ and $\mathcal{S}_1$,
2. obtaining the steady state probabilities of the aggregated process and,
3. obtaining the performance measure, e.g., expected response time or expected number of customers.

The larger the state space cardinality of $\mathcal{S}_i$, the more accurate are the results obtained. In this section, we discuss how we can reduce the state space of $\mathcal{S}_i$ by lumping *similar* states.

Kemeny and Snell [KS60] studied under what conditions an aggregated process is still Markovian. The condition for a Markov process to be lumpable with respect to a partition $\{\mathcal{P}_0 \cup \mathcal{P}_1 \cup \cdots\}$, where $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$, is that for every pair of sets $\mathcal{P}_i$ and $\mathcal{P}_j$, $r_{k,\mathcal{P}_j}$ has the same value for every state $k \in \mathcal{P}_i$ where:

$$
r_{k,\mathcal{P}_j} = \sum_{l \in \mathcal{P}_j} q_{k,l} \qquad \qquad for\ k \in \mathcal{P}_i
$$

We can apply this notion to our minimum expected delay routing problem.

Let J be the number of distinct types of servers in the model where two servers are of the same type if and only if they have the same service rate. For any state s define the following mapping:

$$f : s \quad \rightarrow \quad \{l_i | i = 1, 2, \ldots, J\}$$

where:

l_i = is a set of tuples $(\alpha_{ij}, \beta_{ij})$

α_{ij} = is a queue length for a server of type i that appears in state s

β_{ij} = is the number of servers of type i that has queue length α_{ij} in state s

We define a partition of the state space $\mathcal{S}_u$ (or $\mathcal{S}_l$) by specifying that $s_1, s_2 \in \mathcal{S}_u$(or $\mathcal{S}_l$) are in the same partition if and only if $f(s_1) = f(s_2)$.

For example, assume we have a four server system with $\mu_1 = \mu_2 = 4$, $\mu_3 = 3$ and $\mu_4 = 2$. There are three distinct types of servers and $J = 3$. We can group states such as $s_1 = [3, 4, 2, 1]$ and state $s_2 = [4, 3, 2, 1]$ into the same partition since the $l_i, i = 1, 2, 3$ for both states are:

$$l_1 = \{(4, 1), (3, 1)\}$$

$$l_2 = \{(2, 1)\}$$

$$l_3 = \{(1, 1)\}$$

It is not difficult to see that the condition for lumpability is satisfied and we can greatly reduce the state space of the model that needs to be solved.

5.6 Numerical Examples

In this section, we present two examples to illustrate the bounding algorithm.

The system we consider in our first example consists of four homogeneous servers. To vary the system utilization ρ from 0.1 to 0.9, we fixed the input arrival rate at 4.0 and vary the service rates for all servers. For this example, we set $d = 4$. For $\rho = 0.1$ to 0.7, we set $C_i = 7$, for $\rho = 0.8$, we set $C_i = 9$ and for $\rho = 0.9$, we set $C_i = 10$. Table 5.1 illustrates the upper and lower bound mean response time as a function of system utilization. Percentage error[4] is defined to be $\frac{R_u - R_l}{R_u + R_l}$ x 100%. Note that the bounds are very tight.

System	States	Response Time	Response Time	Spread of	Percentage
Utilization	Generated	Upper Bound	Lower Bound	Bounds	Error
0.1	175	0.100074	0.100074		
0.2	175	0.201692	0.201692		
0.3	175	0.309557	0.309557		
0.4	175	0.431429	0.431429		
0.5	175	0.579080	0.579068	0.000012	0.00103 %
0.6	175	0.773178	0.772967	0.000211	0.01364 %
0.7	175	1.061225	1.056777	0.004448	0.21000 %
0.8	245	1.569928	1.554950	0.014978	0.47931 %
0.9	280	2.867803	2.752649	0.115154	2.04883 %

Table 5.1: Homogeneous servers system

The second system we consider has four heterogeneous servers with $\mu_1 = 10, \mu_2 = 9, \mu_3 = 8$ and $\mu_4 = 6$. To vary the system utilization from 0.1 to 0.9, we fix the service rates for all servers and vary the input arrival rate. We set

[4] if the spread in bounds is less than $< 10^6$, we leave the entries for the spread of the bounds and percentage error blank.

$d = 6$ and for $\rho = 0.1$ to 0.7, we set $\vec{C} =< 9, 8, 7, 5 >$. For $\rho = 0.8$ to 0.9, we set $\vec{C} =< 12, 11, 10, 8 >$. Table 5.2 illustrates the upper and lower bound mean response time.

System Utilization	States Generated	Response Time Upper Bound	Response Time Lower Bound	Spread of Bounds	Percentage Error
0.1	3095	0.103573	0.103301	0.000272	0.13148 %
0.2	3095	0.107718	0.107435	0.000283	0.13153 %
0.3	3095	0.113167	0.112859	0.000308	0.13627 %
0.4	3095	0.120737	0.120305	0.000432	0.17922 %
0.5	3095	0.131729	0.131086	0.000643	0.24466 %
0.6	3095	0.148537	0.147701	0.000836	0.28221 %
0.7	3095	0.176870	0.174620	0.002250	0.64013 %
0.8	6410	0.230285	0.225782	0.004503	0.98735 %
0.9	6410	0.391237	0.372385	0.018852	2.46876 %

Table 5.2: Heterogeneous servers system

To illustrate the tradeoff between computational cost and accuracy of the bounds. Let us consider the homogeneous queueing system in the first example. By fixing the system utilization at 0.9 and increasing the number of states generated, we see the improvement of the bounds in mean response time. The result is illustrated in Table 5.3.

d	C	States Generated	Response Time Upper Bound	Response Time Lower Bound	Spread of Bounds	Percentage Errors
4	7	175	3.157382	2.487368	0.670014	11.86968 %
4	9	245	2.927385	2.624671	0.302714	5.45228 %
4	10	280	2.867803	2.752649	0.115154	2.04883 %
5	12	518	2.790852	2.760358	0.030494	0.54932 %

Table 5.3: Computational Cost vs. Accuracy

5.7 Conclusions

Joining the shortest queue load balancing is appealing not only due to it's simplicity in implementation, but theoretically difficult to analyze. Since the arrival process is state dependent and no close-form solution exits in general. Also due to the fact that each servers has an infinite capacity queue, the state space cardinality of the Markov model is infinite and it becomes impossible to generate all the states space to analyze the Markov model numerically. In this chapter, we have presented an approach to bound the mean response time and the mean number of customer of minimum expected delay routing policy, which is a generalization for join the shortest queue routing policy. Since the original model has infinite state space and without closed-form solution, We showed that by constructing two modified models with finite state space, M_u and M_l, upper and lower bound mean response time and mean number of customers in the system can be obtained with much less cost. The algorithmic approach provides the flexibility to tradeoff computational resources and tighter bounds.

CHAPTER 6

CONCLUSIONS

The study presented in this dissertation was motivated by our interest to evaluate Markov chain models with very large state spaces. In particular, we concentrated on the study on dependability analysis of highly fault-tolerant computer systems, whose Markov models usually have large state space cardinality. We also studied the performance of a load balancing algorithm which use minimum expected delay for routing job to service centers. The corresponding Markov model has an infinite state space.

In chapter 2 of this dissertation, we proposed a methodology for computing bounds on the steady state availability of complex computer systems. We assume the original Markov model to have a block upper Hessenberg form. The bounding methodology allows us to generate part of the transition rate matrix at each step and at each step, tighter bounds on the steady state availability can be obtained.

The block upper Hessenberg assumption in chapter 2 implies that the probability of two or more components becoming operational from failure in an interval of length $\triangle t$ is $o(\triangle t)$. This is clearly not the case for the system where several components form a module which is considered to have failed if certain specified combinations of components are failed. In this case, the repair process will replace the module as a whole and thereby the upper block Hessenberg assumption will not hold. With this in mind, we presented a generalization of the bounding methodology in chapter 4 for system dependability analysis on a Markov process that has a general transition structure. Since the Markov process has a general transition structure, tight bounds cannot be guaranteed in all cases (unless most of the interesting states are generated, which sometimes may be prohibitive due to memory limitations). We also discussed under what situations we can have reasonable error bounds.

For parallel (or distributed) computer systems, we need to evenly utilize all the system resources in order to achieve an acceptable performance. One way to evenly utilize system resources is by using some form of load balancing algorithm (e.g., join the shortest queue algorithm). In chapter 5, we presented an algorithmic approach to bound the mean response time and mean number of customers in a load balancing algorithm which uses minimum expected delay routing policy. This kind of load balancing algorithm is a general case of the join the shortest queue algorithm. The bounding algorithm provides flexibility to tradeoff computational costs and accuracy.

We can pose the following question: Where does this research go from here? It is the author's opinion that as we build larger and more complex computer systems, formal performance evaluation and prediction are extremely necessary in the design process. The advantages of using analytical modeling are precision and cost effectiveness. Also due to the advance in computer technologies, we are now able to analyze much larger models. However, the models state space typically grows exponentially with parameters such as number of servers, number of customers, etc. Due to this explosive growth, computing power will not grow rapidly enough to solve the problem. However, some models have intrinsic properties that we can exploit. For example, a model might be nearly completely decomposable. In this case, we can apply decomposition technique [Cou77] or iterative aggregation and dis-aggregation techniques [Tak72] to analyze the Markov model. On the other hand, some models have an inherent *skewness* property. That is the steady probability mass is concentrated in small subset of states rather than distributed uniformly over the total state space. In this case, we need to study further the following:

1. determination of the most "popular states".
2. specification and generation of an irreducible Markov process based on these popular states.

To automate the steps describe above for general application is certainly not trivial. After we generate the model, we need techniques to proof bounds on performance measure. We believe approaches such as Markovian proof technique [Dij90] or sample path analysis [Sto83] are the powerful tools to prove the performance bounds.

Pertaining to possible extensions to this research, the author believes that it is theoretically interesting to generalize the bounding methodology to a Markov process with general transition structure as well as general reward rates. Although the bounding methodologies presented in this dissertation are couched in terms of availability models and load balancing model, the approach appears to have promise for other applications. As we stated before, many applications possess the skewness property. For example, it is reasonable to expect models for probabilistic protocol evaluations [DC88, MS87] to have this kind of characteristic. One possible extension is to investigate how to choose the parameters d and C_i such that we can a prior predict the error bounds. Another possible extension for bounding mean response time of load balancing algorithm is to relax some conditions, e.g. by allowing general interarrival and/or service distribution.

APPENDIX A

Proof for Theorem 3.4

Theorem 3.4 $\pi_{D/G_3} \leq \pi_{D/G_2}$

Proof : Given that G is a transition rate matrix for a finite, irreducible continuous time Markov chain $\mathcal{G}$. Then $\mathcal{G}$ is uniformizable which means that $\mathcal{G}$ can be transformed to a discrete time Markov chain with transition probability matrix P which has the same stationary probability vector π [Ros83]. This transformation is achieved by:

$$P = I + \lambda^{-1} G$$

where λ is greater than or equal to the largest absolute diagonal element of G. Here, we uniformize the two rate matrices G_2 and G_3 into P_2 and P_3 with $\lambda = max(\lambda_2, \lambda_3)$ where $\lambda_2\,(\lambda_3)$ is the largest of the absolute values of the diagonal elements of $G_2\,(G_3)$.

We define the following notation:

$p_{i,j}$ = the transition probability from state i to state j in P_2.

$p'_{i,j}$ = the transition probability from state i to state j in P_3.

$d_{i,j}$ = $p_{i,j} - p'_{i,j}$.

$r(i)$ = reward (either 0 or 1) for state i.

$T_2[r(i)]$ = one-step expected reward of P_2 given the present state is i, i.e.

$$T_2[r(i)] = \sum_j p_{ij}\, r(j)$$

$T_3[r(i)]$ = one-step expected reward of P_3 given the present state is i, i.e.

$$T_3[r(i)] \;=\; \sum_j p'_{ij}\; r(j)$$

$R_2^k(i)$ = k-step ($k-1$ transitions plus initial position) accumulative reward for P_2 given the initial state is i, i.e.

$$R_1^k(i) \;=\; \sum_{l=0}^{k-1} T^l[r(i)] \qquad\qquad for\ k = 1, 2, 3, \ldots$$

and $T_1^0[r(i)]$ be an identity function.

$R_3^k(i)$ = k-step ($k-1$ transitions plus initial position) accumulative reward for P_3 given the initial state is i.

$1\{c\}$ = an indicator function equal to 1 if the condition c is true, else 0.

Based on the results in [Dij88b, Dij90], expected reward for $\mathcal{G}_2$ is greater than or equal to the expected reward for $\mathcal{G}_3$ iff:

$$(T_2 - T_3)R_2^k(i) \;\geq\; 0 \qquad\qquad \forall i\ and\ \forall k$$

Let ϕ be a $1-1$ mapping from $\mathcal{D}$ to $\mathcal{C}$ which maps each state $s_D \in \mathcal{D}$ to the corresponding state $s_C \in \mathcal{C}$.

Let f be any nonnegative function applied to state i of the Markov chain. Since the only difference between G_2 and G_3 is in the rates out of d', then for any nonnegative function f:

$$\begin{aligned}(T_2 - T_3)f(i) \;=\; 1\{i = d'\}\Bigg\{ &\sum_{s_D \in \mathcal{D}} p_{d',s_D} f(s_D) + p_{d',d'} f(d') + \sum_{a_i \in A} p_{d',a_i} f(a_i) \\ &- \sum_{s_D \in \mathcal{D}} p'_{d',s_D} f(s_D) - p'_{d',d'} f(d') - \sum_{s_C \in C} p'_{d',s_C} f(s_C) \\ &\qquad\qquad\qquad - \sum_{a_i \in A} p'_{d',a_i} f(a_i) \Bigg\}\end{aligned}$$

Since

$$\begin{aligned} p_{d',d'} &= p^{'}_{d',d'} \\ p_{d',a_i} &= p^{'}_{d',a_i} \quad \forall\, a_i \in A \\ p_{d',s_D} &\geq p^{'}_{d',s_D} \quad \forall\, s_D \in \mathcal{D} \end{aligned}$$

it follows that:

$$(T_2 - T_3)f(i) \;=\; 1\{i = d'\}\left\{\sum_{s_D \in \mathcal{D}} d_{d',s_D} f(s_D) + \sum_{s_C \in C} d_{d',s_C} f(s_C)\right\}$$

Since by construction of G_3 from G_2,

$$d_{d',s_D} \;=\; -\, d_{d',\phi(s_D)} \qquad \forall s_D \in \mathcal{D}$$

we have:

$$(T_2 - T_3)f(i) \;=\; 1\{i = d'\}\left\{\sum_{s_D \in \mathcal{D}} d_{d',s_D}[f(s_D) - f(\phi(s_D))]\right\}$$

A sufficient condition for the above expression to be greater than or equal to 0 is:

$$[f(s_D) - f(s_C)] \;\geq\; 0 \qquad \forall s_D \in \mathcal{D} \; and \; s_C = \phi(s_D)$$

Letting the function f be $R^k(i)$, then the condition:

$$(T_2 - T_3)R_2^k(i) \geq 0 \qquad \forall i,\; k$$

is satisfied if:

$$R_2^k(s_D) - R_2^k(s_C) \geq 0 \qquad \forall s_D \in \mathcal{D} \;\; and \;\; \forall k$$

where $s_C = \phi(s_D)$.

The above sufficient conditions can be easily proved by induction, for each choice of $s_D \in \mathcal{D}$ and $s_C = \phi(s_D)$.

For $k = 0$, since $R_2^0(i) = 0$ for any state i, the inequalities hold.

Assume $R_2^k(s_D) - R_2^k(s_C) \geq 0$ for $k \leq n$. For $k = n+1$ we have:

$$\begin{aligned} R_2^{n+1}(s_D) - R_2^{n+1}(s_C) &= \Bigg\{ r(s_D) + \sum_{i_D \in \mathcal{D}} p_{s_D,i_D} R_2^n(i_D) + p_{s_D,d'} R_2^n(d') + \\ &\qquad \sum_{a_i \in A} p_{s_D,a_i} R_2^n(a_i) \\ &\quad - r(s_C) - \sum_{i_C \in C} p_{s_C,i_C} R_2^n(i_C) - p_{s_C,d'} R_2^n(d') - \\ &\qquad \sum_{a_i \in A} p_{s_C,a_i} R_2^n(a_i) \Bigg\} \end{aligned}$$

Since we want to bound the stationary state probabilities for states in $\mathcal{D}$, we assign the following reward:

$$r(x) = \begin{cases} 1 & x = s_{\mathcal{D}} \\ 0 & \text{otherwise} \end{cases}$$

and based on the construction of G_3, we have:

$$\begin{aligned} p_{s_D,d'} &= p_{s_C,d'} & \\ p_{s_D,a_i} &= p_{s_C,a_i} & \forall s_C = \phi(s_D) \;\; and \;\; \forall a_i \in A \\ p_{s_D,i_D} &= p_{s_C,i_C} & \forall s_C = \phi(s_D) \;\; and \;\; \forall i_C = \phi(i_D) \end{aligned}$$

it follows that:

$$\begin{aligned} R_2^{n+1}(s_D) - R_2^{n+1}(s_C) &= r(s_D) \;+\; \Bigg\{ \sum_{i_D \in \mathcal{D}} p_{s_D,i_D} [R_2^n(i_D) - R_2^n(\phi(i_D))] \Bigg\} \\ &\geq 0 \qquad\qquad \square \end{aligned}$$

APPENDIX B

Proof for Theorem 3.5

Theorem 3.5 $\pi_{\mathcal{D}/G_2} \geq \pi_{\mathcal{D}/G_3}$

Proof : We use the approach similar with the above theorem. Again, we define the following notations:

$p_{i,j}$ = the transition probability from state i to state j in P_2.

$p'_{i,j}$ = the transition probability from state i to state j in P_3.

$d_{i,j}$ = $p_{i,j} - p'_{i,j}$.

$r(i)$ = reward (either 0 or 1) for state i.

$T_2[r(i)]$ = one-step expected reward of P_2 given the present state is i.

$T_3[r(i)]$ = one-step expected reward of P_3 given the present state is i.

$R_2^m(i)$ = m-step ($m-1$ transitions plus initial position) accumulative reward for P_2 given the initial state is i and $T_2^0[r(i)]$ be an identity function.

$R_3^m(i)$ = m-step ($m-1$ transitions plus initial position) accumulative reward for P_3 given the initial state is i.

Again, the expected reward for G_2 is greater than or equal to the expected reward for G_3 iff:

$$(T_2 - T_3)R_2^m(i) \;\geq\; 0 \qquad\qquad \forall i \textit{ and } \forall m$$

Let f be any nonnegative function applied to state i of the Markov chain. Since the only difference between G_2 and G_3 are the rates from $\mathcal{D}$ to $\mathcal{F}_k''$ in G_2

are change to the rates from $\mathcal{D}$ to $\mathcal{C}_k^{''}$ in G_3. For any nonnegative function f, we have:

$$\begin{aligned}
(T_2 - T_3)f(i) &= \sum_{l=0}^{k-1} 1\{i \in \mathcal{F}_l\}\left\{p_{i,i}f(i) + \sum_{j \notin \mathcal{F}_i} p_{i,j}f(j) - p_{i,i}^{'}f(i) - \sum_{j \notin \mathcal{F}_i} p_{i,j}^{'}f(j)\right\} \\
&= \sum_{l=0}^{k-1} 1\{i \in \mathcal{F}_l\}\left\{\sum_{j \in \mathcal{F}_k^{''}} p_{i,j}f(j) - \sum_{j \in \mathcal{C}_k^{''}} p_{i,j}^{'}f(j)\right\}
\end{aligned}$$

Since by construction of G_3 from G_2,we have a 1-1 mapping function ϕ which maps a state in $\{\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_k\}$ to a state in $\{\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \mathcal{C}_k\}$. It follows that:

$$(T_2 - T_3)f(i) = \sum_{l=0}^{k-1} 1\{i \in \mathcal{F}_l\}\left\{\sum_{j \in \mathcal{F}_k^{''}} \{p_{i,j}(f(j) - f(\phi(j))\}\right\}$$

The sufficient conditions for the above equation to be non-negative are:

$$[f(i) - f(j)] \geq 0 \qquad \forall i \in \mathcal{F}_k^{''},\ j = \phi(i) \in \mathcal{C}_k^{''}$$

Letting the function f be $R^m(i)$, then the condition:

$$(T_2 - T_3)R_2^m(i) \geq 0 \qquad \forall i,\ m$$

is satisfied if:

$$R_2^m(i) - R_2^m(j) \geq 0 \qquad \forall i \in \mathcal{F}_k^{''},\ j = \phi(i) \in \mathcal{C}_k^{''} \ \ and\ \ \forall m$$

The above sufficient conditions can be easily proved by induction. For each choice of $i \in \mathcal{F}_l^{''}$ and $j = \phi(i) \in \mathcal{C}_k^{''}$.

For $m = 0$, since $R_2^0(i) = 0$ for any state i, the inequalities hold.

Assume $R_2^m(i) - R_2^m(j) \geq 0$ for $m \leq n$. For $m = n + 1$, we have:

$$R_2^{n+1}(i) - R_2^{n+1}(j) = \left\{r(i) + p_{i,i}R_2^n(i) + \sum_{s \in \mathcal{F}_{k-1}} p_{i,s}R_2^n(s)\right.$$

$$
\begin{aligned}
&\qquad\qquad\qquad\qquad + \sum_{s\in\{\mathcal{F}_{k+1}\cup\cdots\cup\mathcal{F}_n\}} p_{i,s}R_2^n(s) \\
&- r(j) - p_{j,j}R_2^n(j) - \sum_{s\in\mathcal{C}_{k-1}} p_{j,s}R_2^n(s) \\
&\qquad\qquad\qquad\qquad \left. - \sum_{s\in\{\mathcal{F}_{k+1}\cup\cdot\cdot\mathcal{F}_n\}} p_{j,s}R_2^n(s) \right\}
\end{aligned}
$$

Since we want to bound the steady state probabilities in $\mathcal{D} = \{\mathcal{F}_0 \cup \cdots \cup \mathcal{F}_k^{'}\}$, we assign the following reward:

$$
r(i) = \begin{cases} 1 & i \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases}
$$

and based on the construction of G_3, we have:

$$
p_{i,s} \;=\; p_{j,s} \qquad\qquad for\ j = \phi(i)\, and \forall s
$$

it follows that:

$$
\begin{aligned}
R_2^{n+1}(i) - R_2^{n+1}(j) &= r(i) + p_{i,i}\left\{R_2^n(i) - R_2^n(j)\right\} + \left\{ \sum_{s\in\mathcal{F}_{k-1}} \left(R_2^n(s) - R_2^n(\phi(s))\right) \right\} \\
R_2^{n+1}(i) - R_2^{n+1}(j) &\geq 0
\end{aligned}
$$

References

[Avr90] Alberto Avritzer. "Dynamic Load Sharing Algorithms in Asymmetric Distributed Systems." Technical Report CSD-900023, UCLA, 1990.

[BF88] Richard L. Burden and J. Douglas Faires. *Numerical Analysis.* PWS-KENT Publishing Company, 1988.

[BFS75] R.E. Barlow, J.B. Fussell, and N.D. Singpurwalla. *Reliability and Fault Tree Analysis.* SIAM, Philadelphia, 1975.

[Bla87] J.P.C. Blanc. "A Note on Waiting Times in Systems with Queues in Parallel." *Journal of Applied Probability*, **24**:540–546, 1987.

[CB83] J.W. Cohen and O.J. Boxma. *Boundary Value Problems in Queueing System Analysis.* North Holland, 1983.

[CDL81] A. Costes, J.E. Doucet, C. Landrault, and J.C. Laprie. "SURF: A Program for Dependability Evaluation of Complex Fault-Tolerant Computing Systems." In *Proceeding of Fault-Tolerant Computer Systems*, pp. 72–78, 1981.

[CF86] J.A. Carrasco and J. Figueras. "METFAC: Design and Implementation of a Software Toll for Modeling and Evaluation of Complex Fault-Tolerant." In *Proceeding of Fault Tolerant of Computer Systems*, pp. 424–429, 1986.

[CG87] A.E. Conway and A. Goyal. "Monte Carlo Simulation of Computer System Availability/Reliability Models." In *Proceeding of Fault-Tolerant Computer Systems*, 1987.

[Con84] B.W. Conolly. "The Autostrada Queueing Problem." *Journal of Applied Probability*, **21**:394–403, 1984.

[Cou77] P. J. Courtois. *Decomposability - queueing and computer system applications.* Academic Press, New York, 1977.

[CS84] P.J. Courtois and P. Semal. "Bounds for the Positive Eigenvectors of Nonnegative Matrices and for Their Approximations." *Journal of ACM*, pp. 804–825, October 1984.

[CS85] P.J. Courtois and P. Semal. "On Polyhedra of Perron-Frobenius Eigenvectors." *Linear Algebra and its Application*, **65**:157–170, 1985.

[CS86] P.J. Courtois and P. Semal. "Computable Bounds for Conditional Steady-State Probabilities in Large Markov Chains and Queueing Models." *IEEE Journal on Selected Areas in Communications*, **4**(6), September 1986.

[DC88] D.D. Dimitrijevic and M. Chen. "An Integrated Algorithm for Probabilistic Protocol Verification and Evaluation." Technical Report RC 13901, IBM Tech. Report, 1988.

[Dij88a] Nico M. van Dijk. "Bounds for the call congestion of finite single-server exponential tandem queues." *Operations Research*, **36**:470–477, 1988.

[Dij88b] Nico M. van Dijk. "Simple Bounds for Queueing Systems with Breakdowns." *Performance Evaluation*, **8**:117–128, 1988.

[Dij89a] Nico M. van Dijk. *Queueing Networks with Blocking*, chapter A simple bounding methodology for non-product-form queueing networks with blocking, pp. 3–17. North-Holland, 1989.

[Dij89b] Nico M. van Dijk. "Simple Bounds and Monotonicity Results for Multi-server exponential tandem queues." *Queueing Systems*, **4**:1–16, 1989.

[Dij90] Nico M. van Dijk. "The Importance of Bias-terms for Error Bounds and Comparison Results." In *First International Workshop on the Numerical Solution of Markov Chains*, 1990.

[EVW80] A. Ephremides, P. Varaiya, and J. Walrand. "A simple dynamic routing problem." *IEEE Transaction on Automatic Control*, **25**, 1980.

[FM77] L. Flatto and H.P McKean. "Two Queues in Parallel." *Communication on Pure and Applied Mathematics*, **30**:255–263, 1977.

[GCS86] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, and K.S. Trivedi. "The System Availability Estimator." In *Proceeding of Fault-Tolerant Computer Systems*, pp. 84–89, July 1986.

[GLT86] A. Goyal, S.S. Lavenberg, and K.S. Trivedi. "Probabilistic Modeling of Computer System Availability." *Annals of Operational Research*, **8**:285–306, 1986.

[Gra80] W.K. Grassmann. "Transient and Steady State Results for Two Parallel Queues." *Omega*, **8**:105–112, 1980.

[GS87] D. Gibson and E. Seneta. "Augmented Truncation of Infinite Stochastic Matrices." *Journal of Applied Probability*, **24**:600–608, 1987.

[GSN89] A. Goyal, P. Shahabuddin, P. Nicola, and V.F. Glynn. "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems." Technical Report RC14772, IBM T.J. Watson Research Lab, 1989.

[GT85] R. Geist and K. S. Trivedi. "Ultra-High Reliability Prediction for Fault-Tolerant Computer Systems." *IEEE Transactions on Computers*, **12**(C-32):1118–1127, December 1985.

[Hal85] S. Halfin. "The Shortest Queue Problem." *Journal Applied Probability*, **22**:865–878, 1985.

[HG87] P. Heidelberger and A. Goyal. "Sensitivity Analysis of Continuous Time Markov Chains Using Uniformization." In *Proceedings of the Second International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, May 1987.

[Kin61] J.F.C Kingman. "Two Similar Queues in Parallel." *Annals of Mathematical Statistics*, **32**:1314–1323, 1961.

[Kin69] J.F.C Kingman. "Markov Population Processes." *Journal of Applied Probability*, **6**, 1969.

[KS60] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Van Nostrand Company, 1960.

[LB84] E.E. Lewis and F. Bohm. "Monte Carlo Simulation of Markov Unreliability Models." In *Nuclear Engineering and Design*, pp. 49–62, 1984.

[Lit67] J.D.C Little. "A Proof of the Queueing Formula $L = \lambda W$." *Operations Research*, **9**:383–387, 1967.

[LM90] John C.S. Lui and R. R. Muntz. "Evaluating Bounds on Steady State Availability from Markov Models of Repairable Systems." In *First International Workshop on the Numerical Solution of Markov Chains*, 1990.

[MA82] S.V. Makam and A. Avizienis. "ARIES 81: A Reliability and Life-Cycle Evaluation Tool for Fault Tolerant Systems." In *Proceeding of Fault-Tolerant Computer Systems*, pp. 266–274, June 1982.

[ML90] R. R. Muntz and John C.S. Lui. "Performance Analysis of Disk Arrays under Failure." In *Proceeding of International Conference of VLDB*, 1990.

[MS87] N.F. Maxemchuk and K. Sabnani. "Probabilistic Verification of Communication Protocols." In *Protocol Specification, Testing and Verification*, volume VII, pp. 307–320. Elsevier, 1987.

[MSG89] R. R. Muntz, E. de Souza Silva, and A. Goyal. "Bounding Availability of Repairable Computer Systems." *IEEE Transactions on Computers*, December 1989.

[Neu81] M. F. Neuts. *Matrix-geometric solutions in Stochastic Models - an algorithmic approach.* John Hopkins University Press, Baltimore, MD, 1981.

[NNH90] V.F. Nicola, M.K. Nakayama, P. Heidelberger, and A. Goyal. "Fast Simulation of Dependability models with General Failure, Repair and Maintenance Process." In *Proceeding of Fault-Tolerant Computer Systems*, pp. 491–498, Newcastle upon Tyne, England, June 1990.

[NP89] R.D. Nelson and T.K. Philips. "An Approximation to the Response Time for Shortest Queue Routing." In *ACM SIGMETRICS*, pp. 181–189, 1989.

[NP90] R.D. Nelson and T.K. Philips. "An Approximation for the Mean Response Time for Shortest Queue Routing with General Interarrival and Service Times." Technical Report RC15429, IBM T.J. Watson Research Lab, 1990.

[PGK88] D.A. Patterson, G. Gibson, and R.H. Katz. "A case for redundant arrays of inexpensive disks (RAID)." In *Proceedings SIGMOD*, pp. 109–116, 1988.

[Ros83] S.M. Ross. *Stochastic Processes.* Wiley Series in Probability and Mathematical Statistics, 1983.

[RP87] B.M. Rao and M.J.M. Posner. "Algorithmic and Approximate Analysis of the Shorter Queue Model." *Naval Research Logistics*, **34**:381–398, 1987.

[SA61] H.A. Simon and A. Ando. "Aggregation of Variables in dynamic systems." *Econometrica*, **29**:636–646, 1961.

[Sch68] P. Schweitzer. "Perturbation Theory and Finite Markov Chains." *Journal of Applied Probability*, **5**:401–413, 1968.

[Sen80] E. Seneta. *Non-negative Matrices and Markov Chains.* Springer Verlag, 1980.

[SG85a] William J. Stewart and Ambuj Goyal. "Matrix Methods in Large Dependability Models." Technical Report RC-11485, IBM Research Report, November 1985.

[SG85b] William J. Stewart and Ambuj Goyal. "Matrix Methods in Large Dependability Models." Technical Report RC-11485, IBM Research Report, November 1985.

[SG86] E. de Souza e Silva and H.R. Gail. "Calculating Cumulative Operational Time Distributions of Repairable Computer Systems." *IEEE Transactions on Computers*, **C-35**(4):322–332, April 1986.

[SG89] E. de Souza e Silva and H.R. Gail. "Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization." *Journal of ACM*, **36**(1), January 1989.

[SHG88] P. Shahabuddin, V.F. Heidelberger, P. Goyal, and P.W. Glynn. "Variance Reduction in mean Time to Failure Simulations." In *1988 Winter Simulation Conference Proceedings*, pp. 491–499, 1988.

[Sin81] D. Singh. *Engineering Reliability.* John Wiley Inc, 1981.

[Sto83] D. Stoyan. *Comparison Methods for Queues and other Stochastic Models.* John Wiley Inc, 1983.

[Tak72] Y. Takahashi. *Some Problems for Applications of Markov Chains.* PhD thesis, Tokyo Institute of Technology, March 1972.

[TDG84] K.S. Trivedi, J.B. Dugan, R.R. Geist, and M.K. Smotherman. "Hybrid Reliability Modeling of Fault-Tolerant Computer Systems." In *Computer and Electrical Engineering*, volume 11, pp. 87–108, 1984.

[Tri82] K.S. Trivedi. *Probability & Statistics with Reliability, Queuing and Computer Science Applications.* Prentice Hall, 1982.

[Var62] Richard Varga. *Matrix Iterative Analysis.* Prentice-Halls, England Cliffs, N.J, 1962.

[Web78] R.R. Weber. "On the optimal assignment of customers to parallel queues." *Journal of Applied Probability*, **15**:406–413, 1978.

[Win77] W. Winston. "Optimality of the shortest line discipline." *Journal of Applied Probability*, **14**:181–189, 1977.