MASTERS THESIS

STOCHASTIC BOUNDS FOR RANDOMIZED LOAD BALANCING

A.S. GODTSCHALK

Supervisors:
Prof.dr.ir. S. C. (Sem) Borst (Eindhoven University of Technology)
Dr. F. (Florin) Ciucu (Telekom Innovation Laboratories)

June, 2012
Acknowledgements

This thesis is the result of my graduation project in completion of the Master of Science program Industrial and Applied Mathematics at the Eindhoven University of Technology. The project is carried out partly at the Eindhoven University of Technology and partly at Telekom Innovation Laboratories in Berlin.

I would like to acknowledge the institutions and individuals that made it possible for me to bring this thesis to a good end. First of all Telekom Innovation Laboratories for giving me the opportunity to come and work in Berlin and for being an inspiring setting for me to work on my thesis. Secondly I would like to thank the Eindhoven University of Technology for the facilities they offered.

There are many individuals I am grateful to. First I thank both my supervisors Sem Borst (Eindhoven University of Technology) and Florin Ciucu (Telekom Innovation Laboratories) for their advices and support throughout the project. They commented on several versions of my thesis, for which I am very grateful. Secondly I would like to thank all the people from Telekom Innovation Laboratories for their hospitality and help during my wonderful stay in Berlin.

Last, I would like to thank my family for their love and support. Furthermore, I am grateful to my friends for all the fun we had.

Ton Godtschalk
Eindhoven, June 2012
Randomized load balancing is a cost efficient policy for job scheduling in parallel server queueing systems whereby, with every incoming job, a central dispatcher randomly polls some servers and selects the one with the smallest queue. By exactly deriving the jobs’ average delay in such systems, in explicit and closed form, Mitzenmacher [18] proved the so-called ‘power-of-two’ result, which states that by randomly polling only two servers yields an exponential improvement in delay over randomly selecting a single server. Such a fundamental result, however, was obtained in an asymptotic regime in the total number of servers, and does not necessarily provide accurate estimates for practical finite regimes with a small to moderate number of servers. In this thesis we obtain stochastic lower and upper bounds on the jobs’ average delay in non-asymptotic regimes, by borrowing ideas for analyzing the particular case of the Join-the-Shortest-Queue (JSQ) policy. Numerical illustrations indicate not only that the obtained bounds are remarkably accurate, but also that the existing exact but asymptotic results can be largely misleading in some finite regimes.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>vi</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Goals</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Thesis Outline</td>
<td>2</td>
</tr>
<tr>
<td><strong>2 Join the Shortest Queue model</strong></td>
<td>3</td>
</tr>
<tr>
<td>2.1 Definition JSQ</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Literature on JSQ</td>
<td>4</td>
</tr>
<tr>
<td>2.3 The generator matrix $Q$</td>
<td>5</td>
</tr>
<tr>
<td>2.4 Approaches to the Join the Shortest Queue model</td>
<td>7</td>
</tr>
<tr>
<td>2.4.1 Threshold $T$ and capacity $C$</td>
<td>7</td>
</tr>
<tr>
<td>2.4.2 Approaches for lower bounds</td>
<td>7</td>
</tr>
<tr>
<td>2.4.3 Approaches for upper bounds</td>
<td>10</td>
</tr>
<tr>
<td>2.4.4 Conclusion</td>
<td>12</td>
</tr>
<tr>
<td><strong>3 Supermarket model</strong></td>
<td>15</td>
</tr>
<tr>
<td>3.1 Definition SQ($d$)</td>
<td>15</td>
</tr>
<tr>
<td>3.1.1 Transition rates when all server lengths are different</td>
<td>16</td>
</tr>
<tr>
<td>3.1.2 General transition rates</td>
<td>16</td>
</tr>
<tr>
<td>3.2 The generator matrix $Q$</td>
<td>17</td>
</tr>
<tr>
<td>3.3 SQ($d$) Lower bound model</td>
<td>18</td>
</tr>
<tr>
<td>3.3.1 Redirecting transitions</td>
<td>19</td>
</tr>
<tr>
<td>3.3.2 The generator matrix $Q$</td>
<td>21</td>
</tr>
<tr>
<td>3.4 SQ($d$) Upper bound model</td>
<td>22</td>
</tr>
<tr>
<td>3.4.1 Redirecting transitions</td>
<td>23</td>
</tr>
<tr>
<td>3.4.2 The generator matrix $Q$</td>
<td>24</td>
</tr>
<tr>
<td>3.5 Conclusion</td>
<td>25</td>
</tr>
</tbody>
</table>
## CONTENTS

4 Precedence pairs
   4.1 Establishing precedence pairs ........................................ 29
   4.2 Set of precedence pairs for the supermarket model .................. 30
   4.3 Proof of precedence pairs for the supermarket model ............... 31
      4.3.1 Proof for \((m, m + e)\) ........................................... 32
      4.3.2 Illustration for \((m, m + e_1 - e_2)\) .............................. 36
   4.4 Proof for the SQ(\(d\)) lower bound model ......................... 37
   4.5 Proof for the SQ(\(d\)) upper bound model .......................... 38

5 Analysis of the SQ(\(d\)) lower and upper bound models .............. 41
   5.1 Model description ..................................................... 42
   5.2 Solution of the model .................................................. 46
      5.2.1 Partitioning of the state space into blocks of states .......... 46
      5.2.2 Number of elements in the blocks of states .................. 47
      5.2.3 The generator matrix \(Q\) ........................................ 51
      5.2.4 The generator matrix \(Q\) for the SQ(\(d\)) lower bound model 53
   5.3 SQ(2) lower bound model with \(N = 3\) and \(T = 2\) .................. 59

6 Numerical results ......................................................... 63

7 Conclusions ............................................................... 69
CHAPTER 1

Introduction

1.1 Motivation

Parallel server queueing systems model a wide range of scenarios related to daily situations, e.g., toll booths, bank tellers, supermarket cashiers, etc., or to computer and communication systems, e.g., multi-processor systems, data centers, etc. Scheduling in these complex systems concerns the assignment of a single server to execute each arriving job. Existing scheduling policies reveal an interesting tradeoff between 1) the optimality of some performance metric, e.g., jobs’ (average) delay, and 2) cost efficiency, e.g., in terms of minimizing the amount of overhead. At one extreme, the policy of (non-)randomly selecting a server has no feedback cost (as communication from the servers to the dispatcher) but conceivably leads to very large delays, and even to instabilities when the selection process is not adequately balanced. At the other extreme, the Join the Shortest Queue (JSQ) policy, whereby the dispatcher sends each job to the server with the shortest queue, minimizes delay but has a very high feedback cost because all servers must report their queue lengths for every job arrival, and thus raises a valid concern regarding practical implementations. This tradeoff is a valid concern for companies like T-Mobile, where this project (partly) has been carried out.

In order to reduce the feedback cost, and yet to keep the delay ‘small’, JSQ has been generalized to the supermarket model (SQ(d)), whereby the dispatcher runs JSQ only for a subset of d randomly sampled servers from the uniform distribution (see Mitzenmacher [18] and Luczak and McDiarmid [15]). Note that SQ(1) reduces to a simple uniform random selection when d = 1, and to JSQ when d = N, where N is the total number of servers. A fundamental qualitative result is that SQ(2) yields an exponential improvement over SQ(1) in terms of delay, yet with a conceivably small overhead cost. This result is known as the ‘power-of-two’ result (see Mitzenmacher [18]).

Despite its apparent simplicity, SQ(d) is very difficult to analyze in terms of the delay metric, even for a classical input with Poisson arrivals and exponential job sizes. In fact, SQ(d) can be exactly analyzed only for d = 1, in which case the problem reduces to the M/M/1 queue for all N independent servers. What makes the problem particularly difficult, when d > 1, is that the generator matrix of an underlying N-dimensional Markov chain (representing, for instance, the number of jobs at each of the servers’ queues) has an irregular structure. For this reason, solutions have been so far developed either in asymptotic regimes or in terms of bounds in particular cases.

An exact solution on the average delay was obtained in an asymptotic regime in the total number of servers, i.e., for $N \to \infty$ (see Mitzenmacher [18]); this solution was instrumental to showing the ‘power-of-two’ result. Lower and upper bounds on delay were obtained for the particular case when d = N, i.e., JSQ. The main idea is to transform
the original Markov chain with the inherent irregular structure into Markov chains with some regular structure (see Adan et al. [1], Lui et al. [17], or Zhao and Grassmann [23]). To get a lower bound, for instance, the transformation consists in redirecting some transitions between the states of the original Markov chain in such a way that the new system is less loaded than the original one. Moreover, the newly formed generator matrix has a periodic structure such that its analysis becomes amenable to matrix-geometric techniques (Neuts [20]). This method to compute lower and upper bounds on delay could be applied to the general $SQ(d)$ model as well.

1.2 Goals

In this thesis we extend methods to obtain a Markov chain with a regular structure for computing lower and upper bounds for the mean waiting time for the $SQ(d)$ model. The extension is not straightforward, but on the contrary, because of a compounded transformation process needed to produce Markov chains with a regular structure. We thus want to provide the first non-asymptotic results for the $SQ(d)$ policy that can be applied in finite regimes with a small to moderate number of servers. A drawback of the obtained bounds, however, is that they are obtained in implicit form, as they are based on matrix-geometric techniques, and are thus unable to provide qualitative insight alike the 'power-of-two' result. However, the goal was to obtain a numerically tractable method to compute stochastic lower and upper bounds for the mean waiting time for the $SQ(d)$ model to illustrate that the asymptotic result from Mitzenmacher [18] can be largely inaccurate (e.g., smaller than the lower bound) in regimes with a small number of servers.

1.3 Thesis Outline

This chapter served as introduction to the thesis, explaining motivation and goals. The rest of this thesis is organized as follows. In Chapter 2 we introduce the JSQ policy and the behavior of this system. We present several approaches to obtain bound models for the JSQ model. Then, in Chapter 3 we introduce the $SQ(d)$ model and its modified versions. In Chapter 4 we will prove these modified models are stochastic lower and upper bound models for the supermarket model. Once we have proven this, we will be able to come up with numerical results. This will be the content of Chapters 5 and 6. In Chapter 7 we conclude the thesis.
Join the Shortest Queue model

In many practical situations we see parallel servers. At these servers we see queues form, e.g. in front of toll booths, bank tellers, and supermarket cashiers. People use different strategies to decide which server to join. A common one is to determine the number of jobs at each server and join the server with the fewest jobs. The queueing model with this strategy is called the Join the Shortest Queue model (JSQ). In this chapter we consider this JSQ model. In this thesis however, we consider a more generalized strategy to join a server. Instead of determining the number of jobs at the servers, this strategy only uses the information of \( d \) servers. These \( d \) servers are chosen at random and the job joins the server among these with the fewest jobs. The queueing model with this strategy is called the Supermarket model (SQ(d), where \( d \) stands for the \( d \) randomly chosen servers). As SQ(d) is a generalized version of the JSQ model, it is useful to explore JSQ first. Note that if we have \( N \) servers and we poll all \( N \) servers, SQ(N) is equivalent to JSQ.

In this chapter we start by describing the JSQ model. Then, we give an overview of the literature on the JSQ model. The literature we review considers the optimality of the JSQ policy, analytical results for the joint probability distribution in case of JSQ with only two servers and both simulations and approximations for the mean waiting time in case of JSQ with more than two servers. We consider the problem for obtaining the equilibrium distribution in case of the JSQ model with more than two servers. We show the transition flow diagram in case there are three servers to illustrate the irregular structure of the generator matrix, which hampers the computation of the steady-state probabilities. In addition, we discuss what is done in the literature to tackle this problem. We present various approaches that are based on rescheduling jobs to obtain lower and upper bound models for the equilibrium probability vector of the JSQ model. We consider the key idea of the various transformation approaches for JSQ and the useability of the methods to analyze SQ(d). In Chapter 3 we extend two of the approaches constructing lower and upper bound models for the equilibrium probabilities for SQ(d). This, together with the subsequent analysis, will be the central topic of this thesis.

### 2.1 Definition JSQ

We start by giving a formal definition of JSQ. In the JSQ model, jobs arrive at a collection of \( N \) servers and each job joins the server with the fewest jobs (see Figure 2.1). We assume a Poisson arrival process, i.e., the interarrival times are exponentially distributed with rate \( \lambda N \). Also we assume jobs are served according to the first-in-first-out (FIFO) discipline, and we assume the service times of jobs are independent and exponentially distributed with rate \( \mu \). For convenience, in this thesis we assume \( \mu \) is unitary and therefore the stability condition is \( \lambda < 1 \).
Throughout this thesis, we describe the queue lengths at the servers by a state. Concretely, the state space is described by the set

\[ \mathcal{M} = \{ m : m = (m_1, m_2, \ldots, m_N) \} , \]

where \( m_1 \) denotes the largest number of jobs at one of the \( N \) servers, \( m_2 \) denotes the second largest number of jobs, and so on, such that \( m_N \) denotes the smallest number of jobs. Throughout this thesis, we often consider the queue length of a server. When we consider the queue length, we consider both the waiting jobs at a server and the job in service. Due to the Poisson arrivals and exponential service times the next state of the system only depends on the current state. Therefore, the system can be described by a continuous-time homogeneous Markov chain.

The transition rates between states are as follows:

- \( \lambda(m, m + e_{N+1-i}) = \lambda N, \text{ with } 1 \leq i \leq N \text{ in case there are } i \text{ servers with the fewest jobs}; \)
- \( \mu(m, m - e_i + j - 1) = j \mu, \text{ with } 1 \leq i \leq N \text{ in case there are } j \text{ servers having the same number of jobs, i.e. } j \text{ servers of length } m_i, \text{ starting from server } i, \)

where \( \lambda(m, m + e_{N+1-i}) \) and \( \mu(m, m - e_i + j - 1) \) are the rates to go from state \( m \) to state \( m + e_{N+1-i} \) and \( m - e_i + j - 1 \), respectively. Here, \( e_i \) is defined as the unit vector containing only zero's except for the \( i' \)th element that is one.

Note that, in case there are \( j \) servers having the same number of jobs, the departure rate \( \mu(m, m - e_i + j - 1) \) is 0 for \( k = i, i + 1, \ldots, i + j - 2 \). We store the transition rates in a generator matrix, which we denote by \( Q \). We want to use this matrix \( Q \) to solve the equilibrium probabilities \( \pi \) from the following system of equations

\[ \pi Q = 0, \pi e = 1. \quad (2.1) \]

Here, (in case there are three servers), \( \pi = (\pi_{(0,0,0)}, \pi_{(1,0,0)}, \ldots) \) is the equilibrium probability vector with \( \pi_{(0,0,0)} \) defined as the equilibrium probability for state \( (0,0,0) \), \( 0 \) the all zero vector and \( e \) the all one vector. Later in this chapter we will study the computation of the steady-state distribution using the system of Equations (2.1). First we give an overview of the literature on JSQ.

### 2.2 Literature on JSQ

The JSQ model has been extensively studied in the literature. Most articles on JSQ consider the first-in-first-out (FIFO) discipline. For this discipline, the optimality of JSQ is considered by several authors. Winston [22] proved that JSQ is optimal for the symmetric shortest queue problem in case the arrival process is Poisson and the service times have an exponential distribution with the same service rate. In the symmetric shortest queue problem the departure process is statistically identical for each server. Weber [21] extends this optimality proof for JSQ to more general arrival and departure processes. He makes no assumption on the arrival process and he assumes the service time of a job is a random variable with a non-decreasing hazard rate. Ephremides et al. [6] prove the optimality of JSQ for the symmetric shortest queue problem in case of having Poisson arrivals and having two exponential servers using an expected cost function. This technique, which we will also use in Chapter 4, can be extended to any finite number of servers to prove optimality of JSQ for an arbitrary number of servers. Note that these proofs only hold when the job sizes are not highly variable. By highly variable job sizes, we mean that the size of the job...
follows a heavy-tailed distribution, i.e., a distribution for which \( P(X > x) \approx x^{-\alpha} \), where \( \alpha \) is a real number between zero and two. In case the job sizes are highly variable JSQ is even far from being the optimal strategy for sending jobs to servers. Crovella [5] shows this by comparing JSQ with various other routing policies including SITA-E. This routing policy for known highly variable job sizes associates a size range to each host such that the total amount of work directed to each host is the same. He shows by simulations that SITA-E outperforms JSQ. For highly variable job sizes with unknown duration, Harchol-Balter [10] proposes a new algorithm that sends all jobs to the same server. If the job is not completed after a designated time limit associated to the server, the job is killed and sent to the next server. Although this seems counter-intuitive, for highly variable job sizes with unknown duration this algorithm outperforms JSQ. As we are interested in the symmetric shortest queue problem with Poisson arrivals and exponential service times, we consider JSQ since the job sizes are not highly variable in this situation.

A common point of interest of the JSQ model is to find out what the various performance measures are. We are, for example, interested in the mean waiting time and the equilibrium distribution. However, finding these performance measures is often difficult. The only exact results for JSQ available in the literature are for two servers. Kingman [11] and Flatto and McKean [7] use generating functions to derive the joint probability distribution of the number of jobs at each server. Boxma and Cohen [4] show that the analysis of the symmetric shortest queue problem can be reduced to that of a Riemann-Hilbert boundary value problem. Last, Adan et al. [3] introduce the compensation approach to obtain an explicit characterization of the equilibrium probabilities. The key idea here is to find the equilibrium probabilities which can be expressed as an infinite sum of product of powers and improving the error on the two boundaries by alternatively adding terms to compensate. Obviously, compensating the error term in one direction should result in a smaller error term in the other direction. Unfortunately, all these methods only work for two servers.

For more than two servers there is no analytical numerical method available yet to compute the various performance measures of interest for the JSQ model. There are some simulation results and some methods to approximate these performance measures though. Grassman [9] simulates JSQ and concludes that the system is stable whenever the traffic intensity \( \rho = \frac{\lambda}{\mu} \) is smaller than one. If \( \rho \) equals one, the expected number of jobs at a server grows with \( \sqrt{t} \) and if \( \rho \) is greater than one, the expected number of jobs at a server grows linearly in time \( t \). Approximating the number of busy servers by a binomial distribution and assuming the jobs are equally divided over the servers, Lin and Raghavendra [14] approximate the mean response time with an error less than 3.5% for up to 64 servers. Nelson and Philips [19] also observe that JSQ is a way to balance the system. Assuming the jobs are equally divided over the servers, they observe that the mean waiting time depends mostly on the shortest server. The queue length of this shortest server is then approximately \( \lceil \frac{m}{N} \rceil \), where \( m \) is the total number of jobs in the system. They report that their approximation of the mean response time has a relative error of less than 2% for systems with at most 16 servers and with a 0 to 0.99 load range. In general though, the numerical methods mentioned do not provide error bounds.

### 2.3 The generator matrix \( Q \)

The reason why it is difficult to obtain a closed-form solution for JSQ with more than two servers is the structure of the transition flow diagram. To gain insight, in Figure 2.2 we present the transition flow diagram of the JSQ model with three servers.

We created the transition flow diagram in Figure 2.2 using states. By state (3,2,1) we mean there are 3 jobs at the server with the most jobs, 2 jobs at the server with the second most jobs and 1 job at the server with the fewest jobs. To structure these states, we positioned the states with the same total number of jobs in the system under each other. However, the transition flow diagram still has an irregular structure. By irregular we mean there is no clear way to describe the states either by periodicity or recursively. This causes that the generator matrix also has a complex and irregular structure. As mentioned earlier in this chapter, the generator matrix is constructed by the arrival and departure rates in the system. In this thesis we use the variable \( Q \) to denote this generator matrix. Recall we want to use \( Q \) to obtain a closed-form solution for the equilibrium probabilities solving the system of equations as in (2.1).
In principle, it should be possible to solve this system of equations when the Markov chain is homogeneous and irreducible; two conditions that our matrix $Q$ satisfies. However, because the matrix $Q$ is huge and does not have a regular structure it is not possible to solve the system of Equations [2.1]. It is possible though to reduce the complexity of the generator matrix. By reducing the complexity we mean that $Q$ will have a more regular structure. In Section 2.4 we show this regular structure. To reduce the complexity of $Q$ we can use various approaches, but all of the approaches we consider are based on redirecting transitions. The main idea is to redirect some transitions between the states of the original Markov chain in such a way that the amount of work in the system is decreased or increased. Using this technique results in modified models, which will turn out to be lower or upper bound models for JSQ, respectively. In practice, we see such problems arise in applications such as computer network communications and packet-switched data networks (see, e.g., Foschini et al. [8]).

In the next section we list some of these possible approaches to redirect transitions. These approaches have been analyzed by Adan et al. [2], Lui et al. [17] and Zhao and Grassmann [23]. Some of these methods will turn out to be useful for the SQ($d$) model as we will see in Chapter 3.
2.4 Approaches to the Join the Shortest Queue model

In order to circumvent the irregularity problem of the transition flow diagram of JSQ, we want to redirect transitions. Recall that by irregular we mean that the states can not be described by periodicity or recursively. Redirecting transitions in a consistent way will help us to obtain a regular structure for the transition flow diagram of JSQ. We prove the modified models are stochastic lower and upper bound models for JSQ in Chapter 3. In this section, we demonstrate various approaches to obtain these bounds. First we introduce two new parameters used in the various approaches. Then we present the key idea of each approach, show the regular structure of the transition flow diagram of JSQ using the approach, observe what could be possible obstacles for using this approach in case of the supermarket model and suggest solutions to these obstacles. Finally, we discuss which techniques we want to use to approximate the supermarket model.

2.4.1 Threshold $T$ and capacity $C$

The various approaches to redirect transitions that we observe all make use of one of the two variables we now introduce. We define the threshold parameter $T$, which is some arbitrary, fixed integer. This parameter $T$ does not allow the difference between the server with the longest queue length and the server with the shortest queue length to be too large. Concretely, the threshold $T$ forces the constraint

$$m_1 - m_N \leq T. \quad (2.2)$$

Recall we describe the set of states by $\mathcal{M}$, where $\mathcal{M}$ is defined in Section 2.1. In case a transition causes the violation of Inequality (2.2), this transition will be redirected to another state depending on the approach. As JSQ is a way to balance the system (see e.g. Lin and Raghavendra [14] and Nelson and Philips [19]) we do not expect these redirections will occur often. As a result, we do not expect that the lower and upper bound models for JSQ will differ much from the original JSQ model. This expectation is supported by the numerical results of Adan et al. [1], [2], where there are already close results to the original JSQ model for a threshold value of $T = 3$.

The other variable we introduce here is the capacity $C$. In all of the approaches using $C$, the capacity $C$ is the same for all servers. In the literature there are also results for different capacities for the various servers (see e.g. Lui et al. [17]), but for simplicity we only consider the configuration for which all capacities are equivalent for all servers. The main idea of using the capacity $C$ is to regulate the amount of work at the servers. Once there is too much work, there is a modification to the model that simplifies the analysis of the system. Concretely, when the total amount of work (at a server) exceeds the capacity $C$, the model is changed.

2.4.2 Approaches for lower bounds

Threshold Jockeying

Idea:

When a departure causes the violation of Inequality (2.2), the departure occurs from (one of) the longest queue(s) instead of the shortest queue. This technique of redirecting transitions we call threshold jockeying and is among others studied in the literature by Zhao and Grassmann [23] and Adan et al. [2]. Using threshold jockeying results in a more balanced system after the redirected departure. In Chapter 4 we prove redirecting transitions that balance the system leads to lower bound models for the originally considered model. The main advantage is that using this method of redirecting transitions takes away the irregular structure of the transition flow diagram. For convenience, in Figure 2.3 we show the (regular) transition flow diagram of JSQ using threshold jockeying for three servers and a maximal allowable difference of $T = 2$ between the longest queue and the shortest queue.

Problem for SQ($d$):

In the JSQ policy an arrival always joins the server with the fewest jobs. Therefore, the difference between the longest queue and the shortest queue never grows due to an arrival. In addition, using the JSQ routing policy Inequality (2.2) is never violated due to an arrival. However, this does not hold for the SQ($d$) routing policy. Concretely, when all $d$ polled servers are as long as the longest queue and the difference between the longest queue and
the shortest queue is \( T \), Inequality (2.2) is violated due to this arrival. In other words, when the \( d \) polled servers all have length \( m_1 \) and \( m_1 - m_N = T \), the arrival causes the violation of Inequality (2.2).

Possible solution:
When an arrival causes the violation of Inequality (2.2), i.e. by polling the \( d \) longest queues when \( m_1 - m_N = T \), the arrival is sent to (one of) the shortest queue(s) instead of the longest queue. The used approach is again threshold jockeying and the redirected transition results in a lower bound model.

Good bound:
Considering the regular structure in the transition flow diagram of Figure 2.3 and, as mentioned before, the fact that JSQ is a policy that balances the system, we expect the threshold jockeying approach could be a useful approach to analyze SQ(d). The modification on the arrival process does influence the behavior of the transition flow diagram, but the regular structure remains.

\textit{Capacity Jockeying}

\textit{Idea:}
When all capacities of the servers are reached and a departure occurs, it is taken from (one of) the longest queue(s). This technique is called capacity jockeying. Observe that when all capacities are reached the difference between the longest queue and the shortest queue is either zero or one. If all queues are of equal length, i.e. \( m_1 - m_N = 0 \), a departure is taken from server \( N \) and an arrival joins server 1. When there is a difference between the longest queue and the shortest queue, i.e. \( m_1 - m_N = 1 \), the state \( m \) is of the form

\[
\begin{pmatrix}
(m_1, m_1, m_1-1, \ldots, m_1-1) \\
\end{pmatrix}
\]

In this state there are \( c \) servers of length \( m_1 \) and \( N - c \) servers of length \( m_1 - 1 = m_N \) and a departure takes place at server \( c \) and an arrival joins server \( c + 1 \). In fact, if the total number of jobs in the system is such that all capacities of the servers are reached, i.e. \( \#m > NC \), where \( \#m \) stands for the total number of jobs in the system, the system behaves like an \( M/M/1 \) system with arrival rate \( \lambda N \) and departure rate \( \mu N \). This lasts until there are not more than \( NC \) jobs in the system. From that point, the system behaves like the JSQ model again. The transition flow diagram looks very structured from the point the capacities are reached, as can be observed from Figure 2.4 In Figure 2.4 the transition flow diagram of the JSQ model using capacity jockeying is displayed in case of having \( N = 3 \) servers and server capacities \( C = 2 \). Note that when all the capacities are reached, i.e. after state \( (2, 2, 2) \), the difference between the longest and the shortest queue is either zero or one.

Figure 2.3: Transition flow diagram of JSQ with threshold jockeying for \( N = 3 \) and \( T = 2 \)
2.4. APPROACHES TO THE JOIN THE SHORTEST QUEUE MODEL

Problem for SQ(d):
The JSQ routing policy causes the regular structure of the transition flow diagram in Figure 2.4. The key advantage of JSQ when all capacities are reached is that all states have the property that the difference between the longest queue and the shortest queue is either zero or one. This does not hold for SQ(d). Also, in JSQ there is only one transition from a state \( m \) with not all capacities reached to a state \( m' \) with all capacities reached. Concretely, the only transition from a state \( m \) with not all capacities reached to a state \( m' \) with all capacities reached is when an arrival occurs in state \((C,\ldots,C,d)\). Again, this does not hold for SQ(d).

Possible solution:
For both the problem of not having a maximum allowable difference between the longest queue and the shortest queue of one in SQ(d) as for the problem how SQ(d) does not uniquely reach the state with all capacities reached, the arrival that causes this problem should join (one of) the shortest queue(s). The redirected arrival effects in a more balanced system, which lowers the mean waiting time. We prove this in Chapter 4. Note that when all capacities are reached, by redirecting the transitions as we just described results the arrivals always join the shortest queue. Therefore, the routing policy is equivalent to the JSQ routing policy again.

Good bound?:
We expect that it needs a lot of redirected transitions to reach the state where all capacities are reached. When all these capacities are reached, the arrivals do not follow the SQ(d) routing policy anymore and we expect a lot of arrivals that should be redirected. Therefore, we do not expect this bound will help us finding an accurate bound to the SQ(d) model.

Full Service Phase
Idea:
When there are more than \( N \cdot C \) jobs in the system, the system enters a full service phase, where it behaves as a heterogeneous \( M/M/N \) system in which, if there are \( j \) jobs, where \( j < N \) these \( j \) jobs are all executed on different servers. The system operates in this mode until the system becomes idle. Once the system empties, it returns to the normal service mode. Note that for JSQ the only situation in which it enters the full service phase is when an arrival occurs in state \((C,\ldots,C,d)\). This results in a lower bound model for JSQ, as is proven in Lui et al. [17]. Of course, the quality of the bound depends on how large the parameter \( C \) is chosen. The larger \( C \) the better the modified model behaves like the original model. However, this comes at the cost of the efficiency of approximating the model. This results in a tradeoff between on one hand the solvability of the system and on the other hand the spread in the obtained bounds. The irregularity problem for the transition flow diagram is tackled though as we can see in Figure 2.5 where the transition flow diagram is shown for a maximum of \( N \cdot C = 6 \) jobs in the system and \( N = 3 \) servers.
Figure 2.5: Transition flow diagram of JSQ with full service phase for $N = 3$ and $NC = 6$

Problem for SQ(d):
In contrast to JSQ, in SQ(d) there are more states in which the system can enter the full service phase. Therefore, the results for the various performance metrics after the full service mode could differ much from the original SQ(d) model. For example, when an arrival occurs in state $(\frac{NC}{2}, \frac{NC}{2}, 0, \ldots, 0)$ (which, in contrast to JSQ, in SQ(d) is an existing state) it enters the full service phase. The full service phase will empty the system much faster as the original model would do in this situation. Of course, this results in a loose approximation to the original model. An even larger problem seems to be the tradeoff of how large $N \cdot C$ should be. Again as a result of having more states in which the system can enter the full service mode, the analysis of the SQ(d) model using the full service phase approach is compounded compared to the JSQ model. Therefore, the maximum number of jobs in the system cannot be as large for SQ(d) as it was for JSQ. This results in a more loose approximation for SQ(d); an approximation that compared to the approximation of JSQ already was more loose.

Possible solution:
In order to circumvent the problem of having many possible states in which the system can enter the full service phase, we add a capacity $C$ to each server. When an arrival causes the exceedance of the capacity $C$ at a server, the arrival is redirected to (one of) the shortest queue(s). This way, the only situation in which the system enters the full service mode is again when an arrival occurs in state $(C, C, \ldots, C)$. The advantage of using this approach is that it reduces the complexity of solving the modified model. Therefore the parameter $N \cdot C$ could be chosen larger to obtain a better approximation. The disadvantage of using this approach is that the approximation could be more loose due to the redirected transitions.

Good bound:
As we expected for the capacity jockeying approach, we expect there will be a lot of redirections before reaching the state $(C, C, \ldots, C)$. Together with the approximation error due to the full service phase, we expect the lower bound model using the full service approach for SQ(d) will turn into a very loose bound. Therefore, we will not use this approach to approximate the supermarket model.

2.4.3 Approaches for upper bounds

Threshold Blocking
Idea: When a departure causes the violation of Inequality (2.2), the departure may not occur. Instead it starts its service all over again. The technique we use here is called Threshold blocking and has been analyzed by Adan et al. [2]. Blocking these departures causes higher performance characteristics and therefore results in an upper bound model for JSQ. The transition flow diagram looks regular as is demonstrated in Figure 2.6, where the transition flow diagram is shown for JSQ using threshold blocking with $N = 3$ servers and threshold $T = 2$. 
2.4. APPROACHES TO THE JOIN THE SHORTEST QUEUE MODEL

Problem for SQ(d):
As we have seen for the threshold jockeying approach, with the JSQ policy an arrival always joins the server with the fewest jobs. Therefore, the difference between the longest queue and the shortest queue never grows due to an arrival. In addition, Inequality 2.2 is never violated due to an arrival using the JSQ routing policy. However, using the SQ(d) routing policy, an arrival can cause the violation of Inequality 2.2. Concretely, when all d polled servers are as long as the longest queue and the difference between the longest queue and the shortest queue is T, Inequality 2.2 is violated due to an arrival. In other words, when the d polled servers all have length m_1 and m_1 - m_N = T, the arrival causes the violation of Inequality 2.2.

Possible solution:
When an arrival causes the violation of Inequality 2.2, the arrival is accompanied by the addition of one extra job at each of the shortest queues. This technique is called threshold addition and has been analyzed before by Adan et al. 2. The transition flow diagram will still have a regular structure and could therefore be a useful way to obtain an upper bound model for SQ(d).

Good bound?
As JSQ is a way to balance the system (see e.g. Lin and Raghavendra [14] and Nelson and Philips [19]), we expect SQ(d) is also a policy that balances the system. Therefore, we do not expect there will be a lot of blocked departures nor a lot of arrivals that are joined by extra arrivals at each of the shortest queues. Using the threshold blocking together with the threshold addition looks like a promising approach to obtain an upper bound model for SQ(d).

Capacity Blocking
Idea:
When all the capacities of the servers are reached and an arrival occurs, all "active jobs" become "suspended jobs" and they will be in service again when all servers are idle. The system does not take the suspended jobs into account until all servers are empty again. Once the system is empty again, the suspended jobs become active and, as a result, all N servers will be back in service. The technique we use is called capacity blocking. Note that for JSQ it is only possible that an arrival causes the exceedance of the capacities and that this exceedance only happens in some particular states. The first situation (starting with an empty system) where "active jobs" become "suspended jobs" is when an arrival occurs in the state (C, C, ..., C). After this arrival there will be NC suspended jobs and one active job. In this new situation it is (after some time) again possible that the capacities will be exceeded by an arrival. This situation occurs when there is an arrival in state (2C, 2C, ..., 2C). Then there will be NC more suspended jobs leading to a total of 2NC suspended jobs. Note also that the only possible transition where suspended jobs return into active jobs is when there is a departure in the state (Ck + 1, Ck, ..., Ck), where k ∈ N. Here the system returns to state (Ck, Ck, ..., Ck) and all servers become active again. We can repeat this process and this system can be...
solved (see Lui et al. [17]) using matrix-geometric analysis (see Neuts [20]). The key idea here is again that the irregular structure of the transition flow diagram is regulated by the new approach. To gain insight, in Figure 2.7 the transition flow diagram is shown for the capacity blocking approach with $N = 3$ servers and a capacity of $C = 2$ jobs for each server.

![Transition flow diagram of JSQ with capacity blocking for $N = 3$ and $C = 2$.](image)

Figure 2.7: Transition flow diagram of JSQ with capacity blocking for $N = 3$ and $C = 2$

Problem for SQ($d$):
An important property of the JSQ model in the capacity blocking approach is that the only way to exceed all the capacities of the servers is when there is an arrival at a state $(Ck, Ck, \ldots , Ck)$, where $k \in \mathbb{N}$. Starting with an empty system, it is impossible to reach a state with a server having $C + 1$ jobs without passing state $(C, C, \ldots , C)$. This feature of the JSQ routing strategy does not occur in the SQ($d$) routing strategy. Using the SQ($d$) policy it is actually very likely an arrival is sent to a server whose capacity has already been reached, while not for all servers the capacity has been reached yet. Therefore, the irregularity problem for the transition flow diagram is not solved. Killing this arrival is not an option though, since killing an arriving job results in lower performance metrics while we are working on an upper bound model. The arriving job that causes the capacity to be exceeded at one of the servers while not all capacities are already reached, should be redirected in a way that results in strengthening the restrictions on the JSQ model.

Possible solution:
The arriving job that causes the exceedance of a capacity while not being in a state $(Ck, Ck, \ldots , Ck)$ with $k \in \mathbb{N}$, is a big problem. This job can not be redirected to a server having fewer jobs as this results in loosening the restrictions instead of strengthening the restrictions on the JSQ model. Turning the arriving job into a suspended job, causes trouble trying to regulate the transition flow diagram. Doing this would result in an extra variable for the set of states, where for each state $m = (m_1, m_2, \ldots , m_N)$ for the new state $m = (m_1, m_2, \ldots , m_N, m_{N+1})$ all values for $m_{N+1}$ are possible. Turning an arriving job into a suspended job while adding jobs until all the capacities of all servers are reached, i.e. sending the system to state $(Ck + 1, Ck, \ldots , Ck)$, is a possibility. The transition flow diagram will look structured and the adding of jobs will turn the model into an upper bound model. Intuitively, this bound will be a very loose bound though.

Good bound?:
As we discussed in getting a solution for the problem an arriving job can cause, we do not expect this method will help us with analyzing the supermarket model.

### 2.4.4 Conclusion

All of the approaches could form nice bounds in case of the JSQ model. Using these approaches in case of the SQ($d$) model requires some extra modifications though; modifications that for some of the approaches could involve seri-
ous problems. The techniques of capacity jockeying, capacity blocking and the full service phase, seem to contain a huge problem keeping the transition flow diagram structured while not letting the bound becoming too loose. However, the approaches using the threshold $T$ look very promising. The approach needs a slight modification as also arrivals can cause the violation of Inequality (2.2), but we expect redirecting these arrivals is not often needed. Also, as we will see in Chapter 5, the transition flow diagram will still have a regular structure.

Because of the benefits the approaches using the threshold value $T$, we have decided to further investigate the supermarket model by means of these threshold approaches. This will be the core contribution of this thesis. In the next chapter, we define both the supermarket model and the modified models that lead to lower and upper bound models for the supermarket model.
CHAPTER 3

Supermarket model

The key advantage of the JSQ strategy, reviewed in the previous chapter, is that it minimizes the mean waiting time of jobs. This optimality property, however, comes at the cost of a substantial amount of feedback needed from the system, i.e., the number of jobs must be polled for each server and at every job arrival. In actual implementations, such an extensive feedback can impose severe overhead on network and computing resources. In order to alleviate such overhead, and yet obtain close to optimal mean waiting times, the JSQ policy has been generalized to the supermarket model policy. In this model, jobs run the JSQ strategy only for a subset of the servers picked at random.

In this chapter we first describe the supermarket model strategy. Then, we reproduce the key ideas for analyzing the JSQ policy through two transformed models in the case of the supermarket model. Concretely, we further transform the JSQ lower and upper bound models using a threshold (see Section 2.4) by artificially decreasing and increasing, respectively, the number of jobs at the servers. We prove the modified models are stochastic bound models in Chapter 4. We point out that the two artificial transformations are constructed in order to be able to analyze, in a tractable manner, the mean waiting time in the supermarket model through lower and upper bounds, respectively; this analysis will be carried out in Chapter 5. The extension of the JSQ lower and upper bound models, together with the subsequent analysis, represents the central topic of this thesis.

3.1 Definition SQ(d)

We consider the following supermarket model with \( N \) servers, denoted by \( SQ(d) \), where \( 1 \leq d \leq N \) is a fixed integer. In this model, jobs arrive according to a renewal process, while job sizes, or the job service times, are sampled from some distribution. Every arriving job polls \( d \) servers at random, according to a uniform distribution without replacement, out of the \( N \) servers. The \( d \) selected servers report the number of jobs in their systems, and the newly arriving job joins the server with the smallest number of existing jobs; ties are resolved arbitrarily (see Figure 3.1). At every server, jobs are served according to the FIFO policy. Note that, in the particular case when \( d = N \), the routing policy in the \( SQ(d) \) model reduces to the JSQ policy. Also note that, in the particular case when \( d = 1 \), the routing policy in the \( SQ(d) \) model reduces to the random policy. When we consider Poisson arrivals and exponential service times, we observe \( N \) independent \( M/M/1 \) queues.

As we pointed out in the previous chapter, even the reduced JSQ policy has been so far analyzed only for an arrival/service model with Poisson arrivals and exponential service times for the jobs. This simplified model enabled the construction of a tractable Markov chain to represent the system. This way, queueing performance metrics of interest (e.g., the mean waiting time) can be numerically obtained. We assume Poisson arrivals with rate \( \lambda N \), and
exponential service times for jobs with rate 1, for the SQ(d) model. We enforce the stability condition \( \lambda < 1 \).

The Poisson/exponential model enables construction of a continuous-time Markov chain to model the evolution of the SQ(d) policy. Concretely, the state space is the set

\[ \mathcal{M} = \{ m : m = (m_1, m_2, \ldots, m_N) \} , \]

where \( m_1 \) denotes the largest number of jobs at one of the \( N \) servers, \( m_2 \) denotes the second largest number of jobs, and so on, such that \( m_N \) denotes the smallest number of jobs.

### 3.1.1 Transition rates when all server lengths are different

In case all servers have a different number of jobs at their server, the transition rates between the states are as follows:

- \( \lambda(m, m + e_i) = \binom{i-1}{d-1} \frac{\lambda}{N} N \), with \( d \leq i \leq N \); 
- \( \mu(m, m - e_i) = \mu \), with \( 1 \leq i \leq N \),

where \( \lambda(m, m + e_i) \) and \( \mu(m, m - e_i) \) is the rate to go from state \( m \) to state \( m + e_i \) and \( m - e_i \), respectively. Although it will not happen frequently that all servers have a different number of jobs, we will explain these transition rates to gain insight. After that we obtain the arrival and departure rates for an arbitrary state. The departure process is trivial and needs no further explanation. For the arrival process we observe that there are \( \binom{N}{d} \) different ways to poll \( d \) servers out of a total of \( N \) servers. Recall that a job joins server \( i \) if server \( i \) is the server with the least jobs of the \( d \) servers that were polled. Concretely, to let an arrival go to server \( i \), one of the \( d \) chosen servers should be server \( i \) and the other \( d - 1 \) chosen servers should be one of the servers 1, 2, ..., \( i - 1 \). Therefore, there are \( \binom{i-1}{d-1} \) different ways for an arrival to join server \( i \). This means the arrival rate is

\[ \lambda(m, m + e_i) = \binom{i-1}{d-1} \frac{\lambda}{N} N . \]

### 3.1.2 General transition rates

Like JSQ, intuitively SQ(d) is a way to divide the jobs equally over the servers. Therefore it will not happen often that all servers have a different number of jobs. In fact, we frequently see two or more servers having the same number of jobs. This influences both the arrival rate and the departure rate. For example, consider the arrival rate of a certain state \( m \) with \( m_1 \geq m_2 \geq \ldots \geq m_{i-1} > m_i = m_{i+1} = \ldots = m_{i+j} > m_{i+j+1} \geq m_{i+j+2} \geq \ldots \geq m_N \) with \( j \) a positive integer. Note that in this situation there are \( j+1 \) servers of equal length, starting from server \( i \). If the fewest number of jobs at a server of the \( d \) chosen servers equals \( m_i \), the arrival is sent to server \( i \). Here, it is not important which of the servers \( i, i+1, \ldots, i+j \) was polled as the server with the fewest jobs. Therefore, the arrival rate equals

\[ \lambda(m, m + e_i) = \sum_{k=i}^{j} \binom{i-1}{d-1} \frac{\lambda}{N} N . \]
3.2. THE GENERATOR MATRIX Q

Note that the arrival rate to go from state $m$ to $m + e_k$ is 0 for $k = i + 1, i + 2, \ldots, i + j$. Observe that the arrival rate is dependent on both the $d$ polled servers and on the number of servers that are of equal length with the fewest jobs of the $d$ chosen servers. Therefore, we do not give the arrival rates for an arbitrary state. We mention the following property of binomials coefficients though:

$$\binom{N}{d} = \binom{N - 1}{d - 1} + \binom{N - 1}{d}.$$  \hfill (3.1)

By using repeatedly Equation (3.1) we see that

$$\sum_{i=d}^{N} \binom{i-1}{d-1} = 1,$$

so the sum of the arrival rates out of one state is indeed $\lambda N$.

An equal number of jobs at different servers also affects the departure rates. Consider again a state of the form $m$ with $m_1 \geq m_2 \geq \ldots \geq m_{i-1} > m_i = m_{i+1} = \ldots = m_{i+j} > m_{i+j+1} \geq \ldots \geq m_N$ with $j$ a positive integer. Again, in this state there are $j + 1$ queues of equal length, starting from queue $i$. If there is a departure at one of the servers $k$, $i \leq k \leq j$, the departure takes place at server $j$. Therefore, the departure rate is

$$\mu(m, m - e_i) = (j + 1)\mu.$$  

Note that, like for the arrival rates, some departure rates are 0. For the departure process it is not possible to go from state $m$ to state $m - e_k$ for $k = i, i + 1, \ldots, i + j - 1$ and these transition rates are therefore 0. Note that the departure process for the SQ($d$) model is equivalent to the departure process for the JSQ model.

3.2 The generator matrix $Q$

The transitions for all the states can be summarized in a generator matrix, which we denote by $Q$. As we have pointed out in the previous chapter, we want to use this matrix $Q$ to solve the equilibrium probabilities from the system of equations as in (2.1). For completeness, we show this system of equations again.

$$\pi Q = 0, \pi e = 1$$

Here, (in case there are three servers), $\pi = (\pi(0,0,0), \pi(1,0,0), \ldots)$ is the equilibrium probability vector with $\pi(0,0,0)$ defined as the equilibrium probability for state $(0,0,0)$, $\mathbf{0}$ the all zero vector and $\mathbf{e}$ the all one vector. Although the existence of a steady-state distribution is guaranteed by the fact that all states of this irreducible Markov chain are positive recurrent, its explicit computation is hampered by the irregular structure of the generator matrix $Q$. By ‘irregular’ we mean that there is no apparent periodic or recursive generic representation for the infinite sized matrix $Q$ that would allow for solving the system of equations as in (2.1).

Recall that the analysis of the JSQ model is also impeded by a similar irregularity problem in the generator matrix structure. In the case of the SQ($d$) model, this problem is compounded due to additional random polling model of the servers. Figure 3.2 illustrates the irregular behavior in the transition flow diagram in the particular case when there are $N = 3$ servers and $d = 2$ servers are polled at random. To show the irregularity problem is compounded for SQ($d$), in Figures 3.3 and 3.4 we only show the arrival rates of the transition flow diagrams of the JSQ and the SQ($d$) model, respectively.

Because of the irregularity problem, there is no numerically tractable method in the literature known yet to solve the system of equations as in (2.1). Mitzenmacher [18] proved the so-called ‘power-of-two’ result, which states that by randomly polling only two servers yields an exponential improvement in delay over randomly selecting a single server. Such a fundamental result, however, was obtained in an asymptotic regime in the total number of servers.
Luczak and McDiarmid [16] prove that the maximum queue length takes at most two values, which are $\ln \ln N / \ln d + O(1)$ when the number of servers $N$ goes to infinity. So again, this result was obtained in an asymptotic regime. For a small to moderate number of servers, these results do not provide good estimations.

In order to circumvent $Q$’s irregularity problem, we next transform the SQ($d$) model into two modified models. The two models will produce lower and upper bounds for the joint steady-state distribution of the number of jobs at the various servers.

### 3.3 SQ($d$) Lower bound model

We want to obtain a regular structure of the state space of the supermarket model. In the original supermarket model we do not have a regular structure of the generator matrix $Q$, which hampers the calculation of the stationary distribution. In fact, there is no numerically tractable method known in literature yet to compute the various performance metrics of interest. We have seen the irregularity problem in the JSQ model in Chapter 2 as well. The way to avoid this problem was to redirect transitions in the original Markov chain such that the newly obtained
system was less or more loaded than the original one. In Section 2.4 we mentioned various approaches to obtain lower and upper bound models for the SQ model. We also discussed whether the described approaches could be useful for the SQ(d) model. We expected the lower bound model using threshold jockeying and the upper bound model using threshold blocking could be useful approaches for SQ(d). In this section we want to introduce both these approaches. First we further transform the JSQ lower bound model using threshold jockeying to obtain the SQ(d) lower bound model makes use of redirecting transitions. By redirecting transitions, we loosen the restrictions on the SQ(d) model. By this we mean that in case a transition is redirected from state \( m \) to state \( m' \), this will result in lower performance metrics of interest, e.g. the mean waiting time. This way the modified model turns out to be a lower bound model for SQ(d), which will be proven in Chapter 4. As we mentioned also in Section 2.4 the technique of threshold jockeying has been analyzed for the JSQ model by Adan.

### 3.3.1 Redirecting transitions

As we mentioned in Section 2.4 the SQ(d) lower bound model makes use of redirecting transitions. By redirecting transitions, we loosen the restrictions on the SQ(d) model. By this we mean that in case a transition is redirected from state \( m \) to state \( m' \), this will result in lower performance metrics of interest, e.g. the mean waiting time. This way the modified model turns out to be a lower bound model for SQ(d), which will be proven in Chapter 4. As we mentioned also in Section 2.4 the technique of threshold jockeying has been analyzed for the JSQ model by Adan.
et al. [1, 2]. In the JSQ lower bound model, redirecting transitions only occurs for departure transitions; for SQ($d$) we also redirect arrival transitions. Concretely, we introduce a threshold parameter $T$ such that, in the transformed Markov chain, the condition as in Inequality (2.2) must hold. For convenience, we repeat the condition here.

$$m_1 - m_N \leq T$$

Intuitively, this inequality enforces that the difference between the longest queue and the shortest queue may not be too large. To enforce this condition we suitably redirect some transitions from the original chain. In particular, to get a stochastic lower bound, we redirect transitions according to the following two rules:

1. When a departure causes the violation of Inequality (2.2), the departure occurs from (one of) the longest queue(s) instead of the shortest queue (see Figure 3.5).

2. When an arrival causes the violation of Inequality (2.2), the arrival is sent to (one of) the shortest queue(s) instead of the longest queue (see Figure 3.6).

Note that in Section 2.4.2 we have seen an equivalent departing situation for the JSQ lower bound model. Also the way the transitions are redirected on both the JSQ lower bound model and the SQ($d$) lower bound model are
3.3. SQ(D) LOWER BOUND MODEL

Figure 3.5: SQ(2) lower bound model, with $N=6$ servers and threshold $T=3$

Figure 3.6: SQ(2) lower bound model, with $N=6$ servers and threshold $T=3$

equivalent. In contrast, this does not hold for the arrival situation. Since an arrival joins the server with the fewest jobs in JSQ, this arrival can not increase the difference between the server with the most jobs and the server with the fewest jobs. Therefore, an arrival never causes the violation of Inequality (2.2) and arrivals are not redirected in the JSQ lower bound model. However, in SQ(d) it is possible an arrival causes the violation of Inequality (2.2). When the $d$ chosen servers are all as long as the longest queue and the difference between the longest queue and the shortest queue is already $T$, Inequality (2.2) is violated due to this arrival. This is not allowed though so for the SQ(d) lower bound model we also have to redirect arrivals.

We now show some properties of the system in terms of the states. The only possibility to cause the violation of Inequality (2.2) is when the current state $m$ has $m_1 - m_N = T$. In this case, it is impossible to have a departure at (one of) the shortest queue(s) or an arrival at (one of) the longest queue(s), i.e. there is no longer a transition to $(m_1, m_2, \ldots, m_{N-1})$ and $(m_1 + 1, m_2, \ldots, m_N)$ respectively. Instead, using our previous defined redirections, the transitions will be sent to $(m_1 - 1, m_2, \ldots, m_N)$ and $(m_1, m_2, \ldots, m_N + 1)$. We assumed here there is only one longest queue; if there are $c$ longest queues the departure is sent to state $(m_1, m_2, \ldots, m_{c-1}, m_c - 1, m_{c+1}, \ldots, m_N)$. Also we assumed there is only one shortest queue; if there are $c$ shortest queues the arrival is sent to state $(m_1, m_2, \ldots, m_{N-c}, m_{N-c+1} + 1, m_{N-c+2}, \ldots, m_N)$.

3.3.2 The generator matrix $Q$

The key advantage is that these transformations eliminate the irregularity of the generator matrix $Q$ and the irregular structure of the transition flow diagram. To illustrate this, in Figure 3.7, we show the transition flow diagram for the SQ(d) lower bound model with $N=3$ servers, $d=2$ choices and threshold $T=2$.

In Figure 3.7 we positioned the states with the same number of jobs in the system under each other. By doing so we see that after an initial stage the process repeats itself. For example, look at the set of states with 5, 6, 7 jobs in the system. The same pattern of transition rates between the corresponding states is at the set of states with 8, 9, 10 jobs in the system. This pattern is also in the set of states with 11, 12, 13 jobs in the system. In fact, this cycle is repeated infinitely many times. For convenience, see Figures 3.8 and 3.9 where we showed the transition flow diagram for the set of states 5, 6, 7 and for the set of states 8, 9, 10. As a result of the regular structure of the new transition flow diagram, the generator matrix $Q$ also obtains a regular structure. In Chapter 5, we show it is now possible to solve the system of equations as in (2.1) and from there to obtain numerical results for the SQ(d) lower
bound model.

![Figure 3.7: Transition flow diagram of SQ(2) lower bound model with N = 3 and T = 2](image1)

![Figure 3.8: Set of states 5, 6, 7 with N = 3 and T = 2](image2)

### 3.4 SQ(d) Upper bound model

In order to analyze the SQ(d) model in an analytical way, we have to obtain a regular structure of the generator matrix $Q$. By using redirecting transitions, we modify the SQ(d) model in such a way that the modified model will have a regular structure. Like we changed the model in the previous section to obtain a lower bound model, we want to change the model in this section to find an upper bound model for SQ(d). When the results for these SQ(d) lower and upper bound models are relatively close to each other, we found a good approximation for the supermarket model. By relatively close we mean that we can say that the performance measure of interest, e.g. the mean waiting time, must be in a certain interval that is particularly small. This interval is constructed by the results of the SQ(d) lower and upper bound models. Intuitively, like JSQ, SQ(d) is a policy that balances the system. Therefore, the violation of Inequality (2.2) and, as a result, the redirection of the transitions do not occur much. When these redirections do not occur often, the system behaves like the normal SQ(d) model but now being numerically tractable. Unlike the last section we now redirect the transitions such that we obtain higher performance characteristics. The modified model we call the **SQ(d) upper bound model**. The technique we use to obtain the SQ(d) upper bound model is the technique of threshold blocking. The technique for threshold blocking was used before for other rout-
3.4. SQ(D) UPPER BOUND MODEL

For example, in Adan et al. [1], [2], threshold blocking is used to obtain an upper bound model for JSQ. We described the results for the JSQ upper bound model in Section 2.4. As we have seen in the previous section for the SQ(d) lower bound model, in the SQ(d) upper bound model we redirect not just departures but also arrivals.

3.4.1 Redirecting transitions

For redirecting departure and arrival transitions we again make use of the threshold $T$. We use $T$ to strengthen the restrictions on SQ(d) resulting in an upper bound model, such that the condition as in Inequality (2.2) must hold. For convenience, we repeat this condition here.

$$m_1 - m_N \leq T$$

To enforce this condition we suitably redirect some transitions from the original chain. In particular, to get a stochastic upper bound, we redirect transitions in the following way:

1. When a departure causes the violation of Inequality (2.2), the departure may not occur (see Figure 3.10).
2. When an arrival causes the violation of Inequality (2.2), the arrival is accompanied by the addition of one extra job at each of the shortest queues (see Figure 3.11).

We see for the SQ(d) upper bound model an equivalent departing situation as for the JSQ upper bound model in Section 2.4. In contrast, this does not hold for the arrival situation. Since an arrival joins the server with the fewest jobs in JSQ, this arrival can not increase the difference between the server with the most jobs and the server with the fewest jobs. Therefore, an arrival never causes the violation of Inequality (2.2) and arrivals are not redirected in the JSQ upper bound model. However, in SQ(d) it is possible an arrival causes the violation of Inequality (2.2).
When the \( d \) chosen servers are all as long as the longest queue and the difference between the longest queue and the shortest queue is already \( T \), Inequality (2.2) is violated due to this arrival. Violating Inequality (2.2) is not allowed though so for the SQ\((d)\) upper bound model we also have to redirect arrivals.

Now we show some properties of the system in terms of the states. The only possibility to violate Inequality (2.2) is when the current state \( \mathbf{m} \) has \( m_1 - m_N = T \). In this case, it is impossible to have a departure at (one of) the shortest queue(s) or an arrival at (one of) the longest queue(s), i.e. there is no longer a transition to \( (m_1, m_2, \ldots, m_N - 1) \) and \( (m_1 + 1, m_2, \ldots, m_N) \) respectively. Instead, using our previous defined redirections, the transitions are sent to \( (m_1, m_2, \ldots, m_N - 1) \) and \( (m_1 + 1, m_2, \ldots, m_N + 1) \). Here we assumed there is only one shortest queue. When there are \( c \) shortest queues the arrival is sent to \( (m_1 + 1, m_2, \ldots, m_N - c, m_{N-c+1} + 1, \ldots, m_{N-1} + 1, m_N + 1) \).

### 3.4.2 The generator matrix

Redirecting the transitions to other states to strengthen the restrictions, has the positive effect that it eliminates the irregular structure of the generator matrix \( Q \) and the irregular structure of the transition flow diagram. For example, see Figure 3.12 where the transition flow diagram for \( N = 3 \) servers, \( d = 2 \) choices and threshold \( T = 2 \) is shown.

In Figure 3.12 we see that in this transition flow diagram we positioned the states with the same number of jobs in the system under each other. We did this to observe a pattern in the transition flow diagram. We observe that after some initial phase the process repeats itself. For example, look at the set of states with 5, 6, 7 jobs in the system. Compare this set of states with the set of states having 8, 9, 10 jobs in the system. We see the same pattern of transition rates between the states. For convenience, see Figures 3.13 and 3.14 where we showed the transition flow diagram for the set of states 5, 6, 7 jobs in the system and the set of states 8, 9, 10 jobs in the system. When we continue looking at the transition flow diagram we see this pattern is repeated over and over again. Using the
3.5 Conclusion

In this chapter, we defined the supermarket model and its modified versions. To obtain a lower bound model, for instance, we redirected transitions in the original Markov chain such that the newly obtained system was less loaded than the original one. In the next chapter we will prove the modified models are indeed lower and upper bound models for the supermarket model. Once we have proven this, we show in Chapter 5 it is now possible to solve the system of equations as in (2.1). From there we obtain numerical results for the SQ(d) lower and upper bound models. This will be the content of Chapters 5 and 6.
Chapter 4

Precedence pairs

Using the SQ(d) lower and upper bound models as we defined in Sections 3.3 and 3.4, we want to analyze various performance characteristics for the supermarket model. In particular, we want to analyze the mean waiting time for the SQ(d) model. Recall that, in principle, it is possible to compute the various performance characteristics of the SQ(d) model, but this is hampered by the irregular structure of the transition flow diagram. This irregular structure was removed by means of the SQ(d) lower and upper bound models by redirecting transitions. Intuitively, this leads to lower and upper bound models, in the sense that the state in the SQ(d) model is stochastically smaller than the state in the SQ(d) upper bound model. Concretely, we mean transitions should be redirected in such a way that for a state \( m \) in SQ(d) and the same state \( m^{UP} \) in the SQ(d) upper bound model the following inequality should hold

\[
P((m_1, m_2, \ldots, m_N) < x) \geq P((m_1^{UP}, m_2^{UP}, \ldots, m_N^{UP}) < x), \quad x \in \mathbb{N}^N.
\]

We need to formally prove this. Therefore, before we consider the analysis of these modified models, we first prove these models are indeed lower and upper bound models for the SQ(d) model. This will be the content of this chapter.

To prove that the modified models using redirecting transitions result in lower and upper bound models for a particular model, we introduce cost functions. Intuitively, a cost function can be seen as the less/more jobs there are in the system, the lower/higher the costs will be. The key idea of the approach we use to bound the model, is that we redirect transitions in such a way that the expected costs are always decreased or increased to obtain a lower or an upper bound model, respectively. This technique has been analyzed by Adan et al. [1] in case of the JSQ model. As cost functions play a crucial role to prove the SQ(d) lower and upper bound models are valid bounds for SQ(d), we first give a formal definition of cost functions. Then, similar to Adan et al. [1] did for JSQ, we further develop the technique using cost functions to prove the validity of the SQ(d) lower and upper bound models.

As we want to compare various performance characteristics of the SQ(d) model, the SQ(d) lower bound model and the SQ(d) upper bound model, we define a different function for the expected \( n \)-period costs for all these models. For the original SQ(d) model, we define \( v_n(m) \) as the expected \( n \)-period costs when starting in a state \( m \in \mathcal{M} \), where \( \mathcal{M} \) was defined as

\[
\mathcal{M} = \{m : m = (m_1, m_2, \ldots, m_N)\},
\]

where \( m_1 \) denotes the largest number of jobs at one of the \( N \) servers, \( m_2 \) denotes the second largest number of jobs, and so on, such that \( m_N \) denotes the smallest number of jobs. Similar to the definition of the expected \( n \)-period costs for SQ(d), we define \( u_n(m) \) and \( w_n(m) \) as the expected \( n \)-period costs starting in state \( m \) for the SQ(d) lower...
and upper bound model, respectively. Intuitively, by \( u_n(m) \) we mean that if we are in state \( m \) and we look \( n \) periods back in time, the expected costs are \( u_n(m) \). Therefore, if we look zero periods back in time, the expected 0-period costs \( u_0(m) \) are zero. This means that we have \( u_0(m) = v_0(m) = w_0(m) = 0 \). Taking this as the start of our induction, we now want to prove by induction that for all (relevant) states \( m \) and for all periods \( n \)

\[
u_n(m) \leq u_n(m) \leq w_n(m).
\]

To establish Inequality (4.1) we only consider

\[
v_n(m) \leq u_n(m),
\]

as the way to prove \( u_n(m) \leq v_n(m) \) is equivalent to the way we prove Inequality (4.2).

To prove Inequality (4.2) we first define precedence pairs. We say that state \( m \) has precedence over, or is more preferable than state \( m' \), if \( m \) and \( m' \) satisfy the following precedence relation

\[
v_n(m) \leq v_n(m'), n \geq 0.
\]

Also, if relation (4.3) holds we say that state \( m' \) is less preferable than state \( m \).

Next we characterize a set \( P \) of pairs \((m, m')\) satisfying Inequality (4.3). All pairs satisfying Inequality (4.3) are called the precedence pairs. In the next section we will see what the precedence pairs are for our modified models. Once a sufficiently rich set \( P \) has been characterized, the proof of Inequality (4.2) follows by induction. We prove this now.

Note that by having \( u_0(m) = v_0(m) = w_0(m) = 0 \), the induction hypothesis holds for \( n = 0 \). Now let \( n \in \mathbb{N} \) and suppose Inequality (4.2) holds for an arbitrary state \( m \). Then the expected costs over \( n + 1 \) periods is

\[
v_{n+1}(m) = c(m) + \sum_{m'} p(m, m') v_n(m'),
\]

where \( c(m) \) is the cost per period in state \( m \) and \( p(m, m') \) is the transition probability to move from state \( m \) to state \( m' \) in the original model.

Provided the modified chain has been constructed by redirecting transitions to less attractive states (i.e. a transition to the state \( m' \) in the original model is redirected to the state \( \tilde{m}' \) in the modified model with \( v_n(m') \leq v_n(\tilde{m}') \)), we have

\[
v_{n+1}(m) = c(m) + \sum_{m'} p(m, m') v_n(m')
\]

\[
= c(m) + \sum_{m' \in S(m')} p(m, m') v_n(m') + \sum_{m' \notin S(m')} p(m, m') v_n(m')
\]

\[
\leq c(m) + \sum_{m' \in S(m')} p^{UB}(m, \tilde{m}') v_n(\tilde{m}') + \sum_{m' \notin S(m')} p^{UB}(m, \tilde{m}') v_n(\tilde{m}')
\]

\[
= c(m) + \sum_{\tilde{m}' \in T(\tilde{m})} p^{UB}(m, \tilde{m}') v_n(\tilde{m}')
\]

\[
\leq c(m) + \sum_{\tilde{m}} p^{UB}(m, \tilde{m}) w_n(\tilde{m})
\]

\[
= w_{n+1}(m),
\]

where \( S(m') \) is defined as the set containing the states \( m' \) in the original model that are redirected to the states \( \tilde{m}' \) in the modified model and \( T(\tilde{m}) \) is defined as the set containing the states \( \tilde{m} \) in the modified model that come from states \( m' \) in the original model that were redirected. Note that the second inequality follows from the induction
4.1. ESTABLISHING PRECEDENCE PAIRS

hypothesis.

So we have now proven Inequality \((4.2)\) and, as a result, we have also proven Inequality \((4.1)\). Note that if we want to prove that a pair \((m, m')\) is a precedence pair, we have to prove that Inequality \((4.3)\) holds for this pair. This only works when the cost function satisfies

\[ c(m) \leq c(m'). \]

This means we have to assume that the costs function \(c(\cdot)\) is a non-decreasing function. As we can see cost functions as the less/more jobs there are in the system, the lower/higher the costs will be, this is a natural assumption to make. Now letting \(N \to \infty\) and considering the long-term average cost per period, this means that

\[ E[c(m_1, m_2, \ldots, m_N)] \leq E[c(m_{1UP}, m_{2UP}, \ldots, m_{NUP})]. \]

Since this holds for any non-decreasing cost function \(c(\cdot)\), this means that state \(m\) is stochastically smaller than \(m_{UP}\), i.e.,

\[ (m_1, m_2, \ldots, m_N) \leq_{st} (m_{1UP}, m_{2UP}, \ldots, m_{NUP}), \]

or equivalently

\[ P((m_1, m_2, \ldots, m_N) < x) \geq P((m_{1UP}, m_{2UP}, \ldots, m_{NUP}) < x), \quad x \in \mathbb{N}. \]

We are now ready to give the following theorem.

**Theorem 1**: Models containing a non-decreasing cost function \(c(\cdot)\) that are modified by redirecting transitions from the state \(m\) in the original model to the state \(m'\) in the modified model such that \((m, m')\) is a precedence pair, result in stochastic upper bound models to the original model, meaning that for all states \(x \in \mathbb{N}\),

\[ P((m_1, m_2, \ldots, m_N) < x) \geq P((m_{1UP}, m_{2UP}, \ldots, m_{NUP}) < x). \]

Note that if we consider the cost function \(c(\cdot)\) as a function that depends on the total number of jobs in the system, we can use Little’s Law to conclude that Theorem 1 could be used to obtain upper bound models for the mean sojourn time of jobs. As a result, Theorem 1 could be used to obtain upper bound models for the mean waiting time of jobs.

In the next section we establish precedence pairs for the supermarket model. For these precedence pairs we prove Inequality \((4.2)\) and we prove that the modified models we obtain for the supermarket model are stochastic lower and upper bounds.

### 4.1 Establishing precedence pairs

Now that we know Theorem 1 holds, we want to use this theorem to obtain the SQ\((d)\) lower and upper bound models. Therefore, we need to characterize which states are more preferable than other states, i.e. we need to characterize a set of precedence pairs \(P\). As we just mentioned, we want to show for all these precedence pairs \((m, m')\) in \(P\) that Inequality \((4.3)\) holds.

Note that there are many states and, as a result, the set of precedence pairs \(P\) can have many elements. Therefore, it could take much effort to prove Inequality \((4.3)\) for all these precedence pairs in \(P\). However, we can select a sufficiently rich subset of \(P\) to prove that our modified model is a lower or an upper bound model to the original model. The subset of \(P\) is sufficiently rich if every state \(m\) of a precedence pair \((m, m')\) in \(P\) can be written as a linear combination of the states \(m'\) of the precedence pairs \((m, m')\) in the subset of \(P\). We call this subset of precedence pairs \(P-\). Then, by using transitivity, it suffices to show Inequality \((4.3)\) for the precedence pairs in \(P-\). By
using transitivity we mean that the operator ≤ is transitive: if \((m, m')\) and \((m', m'')\) satisfy Inequality (4.3) for \(n + 1\), then so does \((m, m'')\) for \(n + 1\). Hence, if we show Inequality (4.3) for all precedence pairs in \(P_−\), we are done.

Once we have characterized the sufficiently rich subset of precedence pairs \(P_−\), we prove by induction on \(n\) that for all states in \(P_−\) Inequality (4.3) holds. As the first step of the induction we take \(n \) equal to one, yielding for Inequality (4.3) that

\[
c(m) \leq c(m').
\]

So the costs function \(c(\cdot)\) should be a non-decreasing function, which we assumed at the beginning of this chapter.

In the next section we give both the set of precedence pairs \(P\) as the sufficiently rich subset of \(P\) for the supermarket model. After that we will finish the induction on \(n\) for the precedence pairs in \(P_−\) to show that Inequality (4.3) holds for all \(n\).

### 4.2 Set of precedence pairs for the supermarket model

As we discussed in the previous section, to obtain the SQ(d) lower and upper bound models we want to construct a set of precedence pairs for the supermarket model. Recall that for a precedence pair \((m, m')\) Inequality (4.3) holds.

To construct a set of precedence pairs we order the states. Concretely, let \(m = (m_1, m_2, \ldots, m_N)\) and \(m' = (m'_1, m'_2, \ldots, m'_N)\) be two states in the supermarket model. Now we assume the set \(P\) of precedence pairs \((m, m')\) consists of all pairs \((m, m')\) satisfying

\[
\begin{align*}
\sum_{i=1}^{N} m_i &\leq \sum_{i=1}^{N} m'_i, \\
m_1 &\leq m'_1, \\
m_1 + m_2 &\leq m'_1 + m'_2, \\
&\vdots \\
m_1 + m_2 + \ldots + m_{N-1} &\leq m'_1 + m'_2 + \ldots + m'_{N-1}.
\end{align*}
\]

Intuitively, Inequality (4.4) can be seen as it is preferable to have less jobs in the system. Obviously, when there are fewer jobs in the system, the costs will be lower. The rest of the inequalities can be considered as it is preferable to have less jobs in the \(j\) longest queues for any \(j = 1, 2, \ldots, N - 1\). This means it is preferable the state is more balanced. In a more balanced state, the expected time it takes before one server becomes idle is longer than in a less balanced state. Therefore, the efficiency of the servers is improved and hence the mean waiting time of jobs is decreased.

In short we can write these inequalities as

\[
\begin{align*}
\sum_{i=1}^{j} m_i &\leq \sum_{i=1}^{j} m'_i, & j = 1, \ldots, N.
\end{align*}
\]

So \(P\) consists of all pairs \((m, m')\) that satisfy Inequality (4.5). As we do not want to prove Inequality (4.3) for all states in \(P\), we define \(P_−\). We define \(P_−\) as the set of pairs for which \(m\) in a precedence pair \((m, m')\) is equal to \(m + e_1, m + e_2, \ldots, m + e_N, m + e_1 - e_2, m + e_2 - e_3, \ldots, m + e_{N-1} - e_N\). Now \(P_−\) is a subset of \(P\) and the inequalities (4.3) for the pairs in \(P_−\) generate (by using transitivity) the inequalities for all pairs in \(P\). To see this we define...
4.3 PROOF OF PRECEDENCE PAIRS FOR THE SUPERMARKET MODEL

\[ d_1 = m'_1 - m_1 \]
\[ d_2 = m'_2 - m_2 \]
\[ \vdots \]
\[ d_N = m'_N - m_N \]

as the differences between the queue lengths. To show that every state \( m' \) in \((m,m')\) can be written as a linear combination of states \( m'' \) in \((m,m'')\), consider

\[ m' = d_1(m,m + e_1 - e_2) \]
\[ + (d_1 + d_2)(m,m + e_2 - e_3) \]
\[ + (d_1 + d_2 + d_3)(m,m + e_3 - e_4) \]
\[ \vdots \]
\[ + (d_1 + d_2 + \ldots + d_{N-1})(m,m + e_{N-1} - e_N) \]
\[ + (d_1 + d_2 + \ldots + d_N)(m,m + e_N). \]

Hence the inequalities (4.3) for the pairs in \( P_- \) generate (by using transitivity) the inequalities for all pairs in \( P \).

Note that

\[ d_1 + d_2 \geq 0 \]
\[ d_1 + d_2 + d_3 \geq 0 \]
\[ d_1 + d_2 + d_3 + d_4 \geq 0 \]
\[ \vdots \]
\[ d_1 + d_2 + \ldots + d_N \geq 0. \]

So we need to show Inequality (4.3) for all pairs \((m,m')\) in \( P_- \) for \( n + 1 \). To do so we need to show for each \((m,m')\) in \( P_- \) that

\[ v_{n+1}(m) = c(m) + \sum_i p(m,i)v_n(i) \]
\[ \leq c(m') + \sum_j p(m',j)v_n(j) \]
\[ = v_{n+1}(m'). \] (4.6)

In the next section we will demonstrate this for some pairs \((m,m')\) in \( P_- \). The proof for other pairs \((m,m')\) in \( P_- \) is equivalent to the proof of the pairs we consider.

4.3 Proof of precedence pairs for the supermarket model

We want to show for all pairs \((m,m')\) in \( P_- \) that Inequality (4.3) holds. We will show Inequality (4.6) for \((m,m + e_N)\). For \((m,m + e_1 - e_2)\) we will partly show Inequality (4.6). The rest of the proof goes in a similar way and needs no further explanation. Also the proof for the other pairs is equivalent to the proofs we consider.
4.3.1 Proof for \((m, m + e_N)\)

To prove Inequality [4.6] holds for \((m, m + e_N)\) we distinguish four cases.

- \(m_N = 0, m_{N-1} = 1\)
- \(m_N = 0, m_{N-1} > 1\)
- \(m_N \geq 1, m_{N-1} = m_N + 1\)
- \(m_N \geq 1, m_{N-1} > m_N + 1\)

Note that when \(m_N = 0\) we must have \(m_{N-1} \geq 1\). For \((m, m + e_N)\) it is useless to consider \(m_{N-1} = 0\), because the state \((m, m + e_N)\) will look like \((m_1, m_2, ..., m_{N-2}, 0, 1)\). Because of the ordering \(m_1 \geq m_2 \geq ... \geq m_N\) this state \((m_1, m_2, ..., m_{N-2}, 0, 1)\) does not exist. Therefore it is impossible to compare \(v_{n+1}(m)\) with \(v_{n+1}(m + e_N)\) for \(m_{N-1} = 0\).

Following the same reasoning we conclude that whenever \(m_N \geq 1\) we must have \(m_{N-1} \geq m_N + 1\).

Case 1: \(m_N = 0, m_{N-1} = 1\)

For convenience, we define the variable \(j\) that has the property that \(m_j = 1\) and \(m_{j-1} > 1\). We want to compare the terms \(v_{n+1}(m)\) and \(v_{n+1}(m + e_N)\) to end up with \(v_{n+1}(m) \leq v_{n+1}(m + e_N)\). We start to express the term \(v_{n+1}(m)\).

\[
v_{n+1}(m) = c(m) + \lambda \sum_{i=1}^{j} p(m, m + e_i) v_n(m + e_i) + \lambda p(m, m + e_N) v_n(m + e_N) + \mu \sum_{i=1}^{j-1} p(m, m - e_i) v_n(m - e_i) + \mu p(m, m - e_{N-1}) v_n(m - e_{N-1}) + \mu v_n(m)
\]  

Because \(m_j = m_{j+1} = ... = m_{N-1} = 1\) and the ordering \(m_1 \geq m_2 \geq ... \geq m_N\), an arrival will never join server \(m_{j+1}, m_{j+2}, ..., m_{N-1}\). This means that \(p(m, m + e_i) = 0\) for all \(j + 1 \leq i \leq N - 1\).

By similar arguments a departure from server \(m_j, m_{j+1}, ..., m_{N-2}\) is also impossible. So \(p(m, m - e_i) = 0\) for all \(j \leq i \leq N - 2\).

The last term in (4.7) corresponds to a departure that actually does not occur. It does not harm our claim though and we will see later that it is useful to show the term.

Next we express the term \(v_{n+1}(m + e_N)\).

\[
v_{n+1}(m + e_N) = c(m + e_N) + \lambda \sum_{i=1}^{j} p(m, m + e_i) v_n(m, m + e_i + e_N) + \lambda p(m, m + e_N) v_n(m + e_j + e_N) + \mu \sum_{i=1}^{j-1} p(m, m - e_i) v_n(m - e_i + e_N) + \mu p(m, m - e_{N-1}) v_n(m) + \mu v_n(m)
\]

We are not interested in the exact expressions for \(p(m, m + e_i)\) and \(p(m, m - e_i)\), although we are able to compute them for all states \(m\). What is important however, is that \(p(m, m + e_i) = p(m + e_N, m + e_i + e_N)\) for all \(1 \leq i \leq j - 1\). This holds since arrivals in states \(m\) and \(m + e_N\) will be sent to server \(i\) with the same probability, because the
longest \( j - 1 \) queues remain the same and are still greater than 1.

An arrival is sent to server \( j \) when (at least) one of the \( d \) choices was one of the servers \( j, j + 1, \ldots, N - 1 \) and the other choice was not server \( N \). This way the shortest queue of the \( d \) choices is one of the queues \( j, j + 1, \ldots, N - 1 \) and the arrival is sent to server \( j \). This holds for both state \( \mathbf{m} \) and state \( \mathbf{m} + \mathbf{e}_N \). So when an arrival was sent to server \( j \) in (4.7), this arrival will also be sent to server \( j \) in (4.8). So we see \( p(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) = p(\mathbf{m} + \mathbf{e}_N, \mathbf{m} + \mathbf{e}_i + \mathbf{e}_N) \) for all \( 1 \leq i \leq j \).

For similar reasons also \( p(\mathbf{m}, \mathbf{m} - \mathbf{e}_i) = p(\mathbf{m} + \mathbf{e}_N, \mathbf{m} - \mathbf{e}_i + \mathbf{e}_N) \) for all \( 1 \leq i \leq j - 1 \).

When an arrival was sent to server \( N \) in state \( \mathbf{m} \), this arrival will be sent to server \( j \) in \( \mathbf{m} + \mathbf{e}_N \).

Recall that the induction hypothesis states that for precedence pairs \( (\mathbf{m}, \mathbf{m}^i) \)

\[
v_n(\mathbf{m}) \leq v_n(\mathbf{m}^i).
\]

In particular the induction hypothesis states

\[
v_n(\mathbf{m}) \leq v_n(\mathbf{m} + \mathbf{e}_i), \quad i = 1, 2, \ldots, N.
\]

Now compare the right-hand sides of equalities (4.7) and (4.8). We see

\[
\begin{align*}
\lambda \sum_{i=1}^j p(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) v_n(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) &\leq \lambda \sum_{i=1}^j p(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) v_n(\mathbf{m}, \mathbf{m} + \mathbf{e}_i + \mathbf{e}_N), \text{IH} \\
\lambda p(\mathbf{m}, \mathbf{m} + \mathbf{e}_N) v_n(\mathbf{m} + \mathbf{e}_N) &\leq \lambda p(\mathbf{m}, \mathbf{m} + \mathbf{e}_N) v_n(\mathbf{m} + \mathbf{e}_j + \mathbf{e}_N), \text{IH} \\
\mu \sum_{i=1}^{j-1} p(\mathbf{m}, \mathbf{m} - \mathbf{e}_i) v_n(\mathbf{m} - \mathbf{e}_i) &\leq \mu \sum_{i=1}^{j-1} p(\mathbf{m}, \mathbf{m} - \mathbf{e}_i) v_n(\mathbf{m} - \mathbf{e}_i + \mathbf{e}_N), \text{IH} \\
\mu p(\mathbf{m}, \mathbf{m} - \mathbf{e}_{N-1}) v_n(\mathbf{m} - \mathbf{e}_{N-1}) &\leq \mu p(\mathbf{m}, \mathbf{m} - \mathbf{e}_{N-1}) v_n(\mathbf{m}), \text{IH} \\
\mu v_n(\mathbf{m}) &\leq \mu v_n(\mathbf{m}).
\end{align*}
\]

We see that all terms on the right-hand side of Equation (4.7) are smaller than or equal to the terms on the right-hand side of Equation (4.8). We can conclude case 1 now by

\[
v_{n+1}(\mathbf{m}) \leq v_{n+1}(\mathbf{m} + \mathbf{e}_N),
\]

which is what we wanted to prove.

**Case 2: \( m_N = 0, m_{N-1} > 1 \)**

We want to compare the terms \( v_{n+1}(\mathbf{m}) \) and \( v_{n+1}(\mathbf{m} + \mathbf{e}_N) \) to end up with \( v_{n+1}(\mathbf{m}) \leq v_{n+1}(\mathbf{m} + \mathbf{e}_N) \). We start to express the term \( v_{n+1}(\mathbf{m}) \).

\[
v_{n+1}(\mathbf{m}) = c(\mathbf{m}) + \lambda \sum_{i=1}^{N-1} p(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) v_n(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) \\
+ \lambda p(\mathbf{m}, \mathbf{m} + \mathbf{e}_N) v_n(\mathbf{m} + \mathbf{e}_N) \\
+ \mu \sum_{i=1}^{N-1} p(\mathbf{m}, \mathbf{m} - \mathbf{e}_i) v_n(\mathbf{m} - \mathbf{e}_i) \\
+ \mu v_n(\mathbf{m})
\]

The last term in Equation (4.9) corresponds to a departure that actually does not occur. It does not harm our claim though and we will see later that it is useful to show the term.

Next we express the term \( v_{n+1}(\mathbf{m} + \mathbf{e}_N) \).
Define \( j \) as the number of jobs for which \( m_j = m_N + 1 \) and \( m_{j-1} > m_N + 1 \). We start to express the term \( v_{n+1}(m) \).

\[
v_{n+1}(m + e_N) = c(m + e_N) + \lambda \sum_{i=1}^{N-1} p(m, m + e_i) v_n(m, m + e_i + e_N) + \lambda p(m, m + e_N) v_n(m + e_N + e_N) + \mu \sum_{i=1}^{N-1} p(m, m - e_i) v_n(m - e_i + e_N) + \mu v_n(m) (4.10)
\]

Like for case 1, we are not interested in the exact expressions for \( p(m, m + e_i) \) and \( p(m, m - e_i) \), although we are able to compute them for all states \( m \). What is important however is that \( p(m, m + e_i) = p(m + e_N, m + e_i + e_N) \) for all \( 1 \leq i \leq N - 1 \). This holds since arrivals in states \( m \) and \( m + e_N \) will be sent to server \( i \) with the same probability, because the longest \( N - 1 \) queues remain the same and are still greater than 1.

For similar reasons also \( p(m, m - e_i) = p(m + e_N, m - e_i + e_N) \) for all \( 1 \leq i \leq N - 1 \).

Recall that the induction hypothesis states that for precedence pairs \((m, m')\)

\[ v_n(m) \leq v_n(m') \]

In particular the induction hypothesis states

\[ v_n(m) \leq v_n(m + e_i), \quad i = 1, 2, \ldots, N. \]

Now compare the right-hand sides of equalities \([4.9]\) and \([4.10]\). We see

\[
c(m) \leq c(m + e_N)
\]

\[
\lambda \sum_{i=1}^{N-1} p(m, m + e_i) v_n(m, m + e_i) \leq \lambda \sum_{i=1}^{N-1} p(m, m + e_i) v_n(m, m + e_i + e_N), \text{IH}
\]

\[
\lambda p(m, m + e_N) v_n(m + e_N) \leq \lambda p(m, m + e_N) v_n(m + e_N + e_N), \text{IH}
\]

\[
\mu \sum_{i=1}^{N-1} p(m, m - e_i) v_n(m - e_i) \leq \mu \sum_{i=1}^{N-1} p(m, m - e_i) v_n(m - e_i + e_N), \text{IH}
\]

\[
\mu v_n(m) \leq \mu v_n(m).
\]

We see that all terms on the right-hand side of Equation \([4.9]\) are smaller than or equal to the terms on the right-hand side of Equation \([4.10]\). We can conclude case 2 now by

\[ v_{n+1}(m) \leq v_{n+1}(m + e_N), \]

which is what we wanted to prove.

**Case 3:** \( m_N \geq 1, m_{N-1} = m_N + 1 \)

Define \( j \) as the number of jobs for which \( m_j = m_N + 1 \) and \( m_{j-1} > m_N + 1 \). We start to express the term \( v_{n+1}(m) \).

\[
v_{n+1}(m) = c(m) + \lambda \sum_{i=1}^{j} p(m, m + e_i) v_n(m, m + e_i) + \lambda p(m, m + e_N) v_n(m + e_N) + \mu \sum_{i=1}^{j-1} p(m, m - e_i) v_n(m - e_i) + \mu p(m, m - e_N-1) v_n(m - e_N-1) + \mu v_n(m - e_N) (4.11)
\]
4.3. PROOF OF PRECEDENCE PAIRS FOR THE SUPERMARKET MODEL

Because \( m_j = m_{j+1} = \ldots = m_{N-1} = 1 \) and the ordering \( m_1 \geq m_2 \geq \ldots \geq m_N \), an arrival will never join server \( m_{j+1}, m_{j+2}, \ldots, m_{N-1} \). This means that \( p(m, m + e_i) = 0 \) for all \( j + 1 \leq i \leq N - 1 \). By similar arguments a departure from server \( m_j, m_{j+1}, \ldots, m_{N-2} \) is also impossible. So \( p(m, m - e_i) = 0 \) for all \( j \leq i \leq N - 2 \).

Next we express the term \( v_{n+1}(m + e_N) \).

\[
v_{n+1}(m + e_N) = c(m + e_N) + \lambda \sum_{i=1}^{j} p(m, m + e_i) v_n(m, m + e_i + e_N)
+ \lambda p(m, m + e_N) v_n(m + e_N)
+ \mu \sum_{i=1}^{j-1} p(m, m - e_i) v_n(m - e_i + e_N)
+ \mu p(m, m - e_N) v_n(m) + \mu v_n(m)
\] (4.12)

Now comparing the right-hand sides of Equations (4.11) and (4.12), we see all terms of (4.11) are smaller than or equal to corresponding terms of (4.12). We can conclude case 3 now by

\[
v_{n+1}(m) \leq v_{n+1}(m + e_N).
\]

**Case 4: \( m_N \geq 1, m_{N-1} > m_N + 1 \)**

First we express \( v_{n+1}(m) \)

\[
v_{n+1}(m) = c(m) + \lambda \sum_{i=1}^{N-1} p(m, m + e_i) v_n(m, m + e_i)
+ \lambda p(m, m + e_N) v_n(m + e_N)
+ \mu \sum_{i=1}^{N-1} p(m, m - e_i) v_n(m - e_i)
+ \mu v_n(m - e_N).
\] (4.13)

Then we express \( v_{n+1}(m + e_N) \).

\[
v_{n+1}(m + e_N) = c(m + e_N) + \lambda \sum_{i=1}^{N-1} p(m, m + e_i) v_n(m, m + e_i + e_N)
+ \lambda p(m, m + e_N + e_N) v_n(m + e_N + e_N)
+ \mu \sum_{i=1}^{N-1} p(m, m - e_i) v_n(m - e_i + e_N)
+ \mu v_n(m - e_N).
\] (4.14)

Now comparing the right-hand sides of Equations (4.13) and (4.14), we see all terms of (4.13) are smaller than or equal to corresponding terms of (4.14). We can conclude case 4 now by

\[
v_{n+1}(m) \leq v_{n+1}(m + e_N).
\] (4.15)

This concludes the proof for \( (m, m + e_N) \). We proved for all possible cases of \( (m, m + e_N) \) that Inequality (4.15) holds, which is what we wanted to prove.

35
4.3.2 Illustration for \((m, m + e_1 - e_2)\)

Recall we want to prove Inequality [4.6], which in this illustration translates to

\[ v_{n+1}(m) \leq v_{n+1}(m + e_1 - e_2). \]

To prove this inequality for \((m, m + e_1 - e_2)\) we distinguish six cases. We only illustrate the proof for case five here. The other cases are similar to the techniques we use in case five and the proof for \((m, m + e_N)\).

- \(m_1 = 1, m_2 = 1\)
- \(m_1 > 1, m_2 = 1\)
- \(m_1 > 1, m_1 = m_2, m_2 = m_3 + 1\)
- \(m_1 > 1, m_1 = m_2, m_2 > m_3 + 1\)
- \(m_1 > 1, m_1 > m_2, m_2 = m_3 + 1\)
- \(m_1 > 1, m_1 > m_2, m_2 > m_3 + 1\)

Note that when \(m_2 = 1\) we must have \(m_3 = 0\). For \((m, m + e_1 - e_2)\) it is useless to consider \(m_3 = 1\), because the state \((m, m + e_1 - e_2)\), i.e., the state \((1, 0, 1, m_4, \ldots, m_N)\) does not exist here.

Following the same reasoning we conclude that \(m_2 \leq m_3 + 1\).

Case 5: \(m_1 > 1, m_1 > m_2, m_2 = m_3 + 1\)

Define \(j\) as the number for which \(m_j = m_2 + 1\) and \(m_{j+1} < m_2 + 1\). We want to compare the terms \(v_{n+1}(m)\) and \(v_{n+1}(m + e_1 - e_2)\) to end up with \(v_{n+1}(m) \leq v_{n+1}(m + e_1 - e_2)\). We start to express the term \(v_{n+1}(m)\).

\[
v_{n+1}(m) = c(m) + \lambda p(m, m + e_2) v_n(m + e_2) \\
+ \lambda p(m, m + e_3) v_n(m + e_3) \\
+ \lambda \sum_{i=j+1}^{N} p(m, m + e_i) v_n(m, m + e_1) \\
+ \mu v_n(m - e_1) \\
+ \mu v_n(m - e_2) \\
+ \mu p(m, m - e_j) v_n(m - e_j) \\
+ \mu \sum_{i=j+1}^{N} p(m, m - e_i) v_n(m - e_i)
\]  

(4.16)

Next we express the term \(v_{n+1}(m + e_1 - e_2)\).

\[
v_{n+1}(m + e_1 - e_2) = c(m + e_1 - e_2) + \lambda p(m, m + e_2) v_n(m + e_1 - e_2 + e_2) \\
+ \lambda p(m, m + e_3) v_n(m + e_1) \\
+ \lambda \sum_{i=j+1}^{N} p(m, m + e_i) v_n(m + e_1 - e_2 + e_i) \\
+ \mu v_n(m - e_2) \\
+ \mu v_n(m + e_1 - e_2 - e_j) \\
+ \mu p(m, m - e_j) v_n(m + e_1 - e_2 - e_j) \\
+ \mu \sum_{i=j+1}^{N} p(m, m - e_i) v_n(m + e_1 - e_2 - e_i)
\]  

(4.17)
Recall that the induction hypothesis states that for precedence pairs \((m, m')\)
\[
v_n(m) \leq v_n(m').
\]
In particular the induction hypothesis states
\[
v_n(m) \leq v_n(m + e_i), \quad i = 1, 2, \ldots, N,
\]
\[
v_n(m) \leq v_n(m + e_1 - e_{i+1}), \quad i = 1, 2, \ldots, N - 1.
\]
Now comparing the right-hand sides of Equations 4.16 and 4.17, we see that all terms of Equation 4.16 are smaller than or equal to corresponding terms of Equation 4.17. We can conclude case 5 now by
\[
v_{n+1}(m) \leq v_{n+1}(m + e_1 - e_2).
\]
The other cases can be proven similarly. To complete the induction step we also have to prove 4.6 for the pairs \((m, m + e_1), (m, m + e_2), \ldots, (m, m + e_{N-1}), (m, m + e_2 - e_3), (m, m + e_3 - e_4), \ldots, (m, m + e_{N-1} - e_N)\). These proofs are similar to the cases we treated so far.

We showed that the precedence pairs for the supermarket model are the pairs for which
\[
\sum_{i=1}^{j} m_i \leq \sum_{i=1}^{j} n_i, \quad j = 1, \ldots, N
\]
holds. The subset \(P_-\) that generates these pairs is the set of pairs \((m, m + e_1), (m, m + e_2), \ldots, (m, m + e_N), (m, m + e_1 - e_2), (m, m + e_2 - e_3), \ldots, (m, m + e_{N-1} - e_N)\). For the two pairs \((m, m + e_N)\) and \((m, m + e_1 - e_2)\) we proved
\[
v_n(m) \leq v_n(m'), \quad n \geq 0
\]
using induction.

Using these precedence pairs we are able to use Theorem 1 Redirecting transitions to more (less) preferable states will provide a stochastic lower (upper) bound model for the supermarket model.

4.4 Proof for the SQ(\(d\)) lower bound model

Now that we have proven that the precedence pairs in case of the supermarket model can result in lower and upper bound models by redirecting transitions to more or less preferable states, we are now ready to prove that the SQ(\(d\)) lower and upper bound models are indeed stochastic bounds. In this section we focus on the SQ(\(d\)) lower bound model.

First recall how the redirections were defined for the SQ(\(d\)) lower bound model in Section 3.3. For redirecting transitions we defined the threshold parameter \(T\) to enforce the condition as in Inequality 2.2. For convenience, we repeat the condition here.
\[
m_1 - m_N \leq T
\]
Whenever a transition causes the violation of Inequality 2.2, we redirect this transition according to one of the following two rules:

1. When a departure causes the violation of Inequality 2.2, the departure occurs from (one of) the longest queue(s) instead of the shortest queue.
2. When an arrival causes the violation of Inequality 2.2, the arrival is sent to (one of) the shortest queue(s) instead of the longest queue.
Whenever \( m_1 - m_N = T \) the modifications cause that there is no longer a transition to \((m_1, m_2, \ldots, m_N - 1)\) and \((m_1 + 1, m_2, \ldots, m_N)\). Instead the transitions are sent to \((m_1 - 1, m_2, \ldots, m_N)\) and \((m_1, m_2, \ldots, m_N + 1)\) respectively. Here we assumed that there is only one longest queue; if there are \( c \) longest queues the departure is sent to state \((m_1, m_2, \ldots, m_{c-1}, m_c - 1, m_{c+1}, \ldots, m_N)\). Also we assumed there is only one shortest queue; if there are \( c \) shortest queues the arrival is sent to state \((m_1, m_2, \ldots, m_{N-c}, m_{N-c+1} + 1, m_{N-c+2}, \ldots, m_N)\).

In the previous section we proved that \((m, m + e_N)\) and \((m, m + e_l - e_{l+1})\) are precedence pairs for all \( i \). This means we can say that \( m \) is less preferable than \( m' \) if

\[
m = m' + c_N e_N + \sum_{i=1}^{N-1} c_i (e_i - e_{l+1}),
\]

(4.18)

where the \( c_i \) are nonnegative integers.

Now, using Equation (4.18), we want to show that the \( SQ(d) \) lower bound model redirects the transitions to more preferable states. We see that if there is only one longest queue \((m_1 - 1, m_2, \ldots, m_N)\) is more preferable than \((m_1, m_2, \ldots, m_N - 1)\) because

\[
(m_1, m_2, \ldots, m_N - 1) = (m_1 - 1, m_2, \ldots, m_N) + e_1 - e_N
\]

\[
= (m_1 - 1, m_2, \ldots, m_N) + \sum_{i=1}^{N-1} (e_i - e_{l+1}).
\]

We see that if there are \( c \) longest queues \((m_1, m_2, \ldots, m_{c-1}, m_c - 1, m_{c+1}, \ldots, m_N)\) is more preferable than \((m_1, m_2, \ldots, m_N - 1)\) because

\[
(m_1, m_2, \ldots, m_N - 1) = (m_1, m_2, \ldots, m_{c-1}, m_c - 1, m_{c+1}, \ldots, m_N) + e_c - e_N
\]

\[
= (m_1, m_2, \ldots, m_{c-1}, m_c - 1, m_{c+1}, \ldots, m_N) + \sum_{i=c}^{N-1} (e_i - e_{l+1}).
\]

Also we see that if there is only one shortest queue \((m_1, m_2, \ldots, m_N + 1)\) is more preferable than \((m_1 + 1, m_2, \ldots, m_N)\) because

\[
(m_1 + 1, m_2, \ldots, m_N) = (m_1, m_2, \ldots, m_N + 1) + e_1 - e_N
\]

\[
= (m_1, m_2, \ldots, m_N + 1) + \sum_{i=1}^{N-1} (e_i - e_{l+1}).
\]

In case there are \( c \) shortest queues \((m_1, m_2, \ldots, m_{N-c}, m_{N-c+1} + 1, m_{N-c+2}, \ldots, m_N)\) is more preferable than \((m_1 + 1, m_2, \ldots, m_N)\) because

\[
(m_1 + 1, m_2, \ldots, m_N) = (m_1, m_2, \ldots, m_{N-c}, m_{N-c+1} + 1, m_{N-c+2}, \ldots, m_N) + e_1 - e_{N-c+1}
\]

\[
= (m_1, m_2, \ldots, m_{N-c}, m_{N-c+1} + 1, m_{N-c+2}, \ldots, m_N) + \sum_{i=1}^{c} (e_i - e_{l+1}).
\]

So we redirect transitions only to more preferable states. Hence, the proof is complete.

The \( SQ(d) \) lower bound model is a stochastic lower bound model for the supermarket model.

### 4.5 Proof for the \( SQ(d) \) upper bound model

Like we proved that the \( SQ(d) \) lower bound model is a stochastic lower bound model for the supermarket model, we want to use this strategy to prove that the \( SQ(d) \) upper bound model is a stochastic upper bound model for the
supermarket model. This means we will prove that the redirected transitions are always redirected to less preferable states.

Therefore, we first recall how the redirections were defined for the SQ(d) upper bound model in Section 3.4. For redirecting transitions we defined the threshold parameter $T$ to enforce the condition as in Inequality (2.2). For convenience, we repeat the condition here.

$$m_1 - m_N \leq T$$

Whenever a transition causes the violation of Inequality (2.2), we redirect this transition according to one of the following two rules:

1. When a departure causes the violation of Inequality (2.2), the departure may not occur.
2. When an arrival causes the violation of Inequality (2.2), the arrival is accompanied by the addition of one extra job at each of the shortest queues.

Whenever $m_1 - m_N = T$ the modifications state there is no longer a transition to $(m_1, m_2, \ldots, m_N - 1)$ and $(m_1 + 1, m_2, \ldots, m_N)$. Instead they will be sent to the states $(m_1, m_2, \ldots, m_N)$ and $(m_1 + 1, m_2, \ldots, m_N + 1)$ respectively. Here we assumed that there is only one shortest queue. When there are be $c$ shortest queues the arrival is sent to $(m_1 + 1, m_2, \ldots, m_{N-c-1}, m_{N-c} + 1, m_{N-c+1} + 1, \ldots, m_N + 1)$.

In Section 4.3 we proved that $(m, m + e_N)$ and $(m, m + e_i - e_{i+1})$ are precedence pairs for all $i$. This means we can say that $m$ is less preferable than $m'$ if

$$m = m' + c_N e_N + \sum_{i=1}^{N-1} c_i (e_i - e_{i+1}),$$

where the $c_i$ are nonnegative integers.

Now, using Equation (4.19), we want to show that the SQ(d) upper bound model redirects the transitions to less preferable states. We see $(m_1, m_2, \ldots, m_N)$ is less preferable than $(m_1, m_2, \ldots, m_N - 1)$ because

$$(m_1, m_2, \ldots, m_N) = (m_1, m_2, \ldots, m_N - 1) + e_N.$$  

Also we see

$$(m_1 + 1, m_2, \ldots, m_N + 1) \text{ is less preferable than } (m_1 + 1, m_2, \ldots, m_N) \text{ because}$$

$$(m_1 + 1, m_2, \ldots, m_N + 1) = (m_1 + 1, m_2, \ldots, m_N) + e_N.$$  

We see that if there are $c$ shortest queues $(m_1 + 1, m_2, \ldots m_{N-c-1}, m_{N-c} + 1, m_{N-c+1} + 1, \ldots, m_N + 1)$ is less preferable than $(m_1 + 1, m_2, \ldots, m_N)$ because

$$\begin{align*}
(m_1 + 1, m_2, \ldots m_{N-c-1}, m_{N-c} + 1, m_{N-c+1} + 1, \ldots, m_N + 1) &= (m_1 + 1, m_2, \ldots, m_N) + e_{N-c} + e_{N-c+1} + \ldots + e_N \\
&= (m_1 + 1, m_2, \ldots, m_N) + \sum_{i=N-c}^{N-1} (i - N + c + 1)(e_i - e_{i+1}) \\
&\quad \quad + (c + 1)e_N.
\end{align*}$$

We used Equation (4.19) to conclude we redirect transitions only to less preferable states. Hence, the proof is complete.

The SQ(d) upper bound model is a stochastic upper bound model for the supermarket model.
CHAPTER 5

Analysis of the SQ(d) lower and upper bound models

Now that we have proven that the SQ(d) lower and upper bound models are stochastic bound models for the original SQ(d) model, we are ready to analyze these models. We want to do so to obtain numerically tractable methods to compute the mean waiting time of customers for these models; methods that are not available for the supermarket model itself.

As we have seen in Sections 3.3 and 3.4, the structure of the transition flow diagrams of the SQ(d) lower and upper bound models are well structured, in contrast to the transition flow diagram of SQ(d). This enables us to compute various performance characteristics by numerically tractable methods, which are in implicit form as they are based on matrix-geometric techniques. Before we describe these techniques, we first consider the relation between certain transition probabilities. In Sections 3.3 and 3.4 we saw a structured transition flow diagram. Therefore we expected a relation between certain transition probabilities. Also we expected a relation between certain steady-state probabilities. We show that both these relations exist in case of the SQ(d) lower bound model. For the SQ(d) upper bound model we observe from the structure of the generator matrix $Q$ that the relation between transition probabilities exists as well. In contrast, there is no direct relation between the steady-state probabilities for the SQ(d) upper bound model.

To demonstrate the relations in case of the SQ(d) lower bound model, we introduce a Markov chain imbedded at arrival epochs. Note that it is important to have single arrivals for the imbedded Markov chain. In case of the SQ(d) lower bound model we have single arrivals, but as a result of the redirected transitions (see Section 3.4) we do not have single arrivals in case of the SQ(d) upper bound model. For the imbedded Markov chain of the SQ(d) lower bound model, we prove that, except for boundary states, $p_{m,m'} = p_{m+1,m'+1}$, where $p_{m,m'}$ is the probability to go from state $m$ to state $m'$. We introduce boundary states as a set of states with less structured behavior as it takes an initial phase to reach the periodic structure. In particular, $p_{m,m'} = p_{m+1,m'+1}$ does not hold for boundary states. Although we can not demonstrate $p_{m,m'} = p_{m+1,m'+1}$ using the imbedded Markov chain for the SQ(d) upper bound model, we observe from the periodic structure of the transition flow diagram and, as a result, the periodic structure of the generator matrix $Q$, that $p_{m,m'} = p_{m+1,m'+1}$ still holds for the SQ(d) upper bound model.

To obtain a regular structure for the generator matrix $Q$, we introduce the concept of blocks of states to partition the state space. By doing so, we are able to use matrix-geometric analysis for both modified models. Then we show our second important relation in case of the SQ(d) lower bound model, where we show that there is a constant, say $\omega$, such that every state, again except boundary states, is related to exactly one other state in the sense that the ratio between their equilibrium probabilities is $\omega$. This constant $\omega$ is equal to $\rho^N$, where $\rho$ is the traffic intensity and
In this chapter, we consider both the SQ(d) lower and upper bound models. However, most of the analysis will consider only the SQ(d) lower bound model. Throughout this chapter, we will clearly indicate which model(s) we are considering. As an illustration, we demonstrate the analysis of the SQ(2) lower bound model with $N = 3$ servers and threshold $T = 2$. We start by describing both the modified models and prove important results for the transition probabilities in case of the SQ(d) lower bound model.

5.1 Model description

In this section, we consider the SQ(d) lower and upper bound models, and point out some important properties of the transition probabilities for the SQ(d) lower bound models. The technique we use is similar to what has been done by Zhao and Grassmann [23], where a similar version of the SQ(d) lower bound model is considered for JSQ.

Recall our definition of the SQ(d) lower and upper bound models. In the SQ(d) model jobs arrive at a collection of $N$ servers. Each job chooses some fixed number of $d$ servers at random from the $N$ servers, and joins the server with the fewest jobs among these. For now, it suffices to assume we have single job arrivals and the interarrival times are identically and independently distributed according to an arbitrary distribution function $A(t)$. We assume the mean interarrival time is equal to $\frac{1}{\lambda}$. Also we assume jobs are served according to the first-in first-out (FIFO) policy, and, unless specified otherwise, we assume the service times of jobs are independent and exponentially distributed with unit mean. In order to obtain more insight in the derivations of the results, we drop the assumption that $\mu$ is unitary and we demonstrate that the result holds for arbitrary $\mu$ as long as the stability condition is still satisfied.

Therefore, we enforce the stability condition $\lambda < 1$. For the SQ(d) upper bound model, this condition is no longer sufficient for stability as redirecting transitions to less preferable states causes more work at the servers. In Section 5.2 we introduce the new stability condition for the SQ(d) upper bound model.

As we have pointed out earlier, the computation of the steady-state distribution is hampered by the irregular structure of the generator matrix $Q$. In order to circumvent this problem we next borrow ideas from the JSQ analysis (see Adan et al. [1]), where the original Markov chain is transformed by suitably redirecting transitions such that the new generator matrix has some regular structure. Concretely, we introduce a threshold parameter $T$ such that, in the transformed Markov chains (one for getting lower bounds and another for getting upper bounds), the condition as in Inequality (2.2) must hold. For completeness, we repeat this condition here.

\[ m_1 - m_N \leq T. \]

Intuitively, this inequality enforces that the difference between the longest queue and the shortest queue will not be too large. To enforce this condition we suitably redirect some transitions from the original chain. In particular, to get a stochastic lower bound, we redirect transitions according to the following two rules:

1. When a departure causes the violation of Inequality (2.2), the departure occurs from (one of) the longest queue(s) instead of the shortest queue.

2. When an arrival causes the violation of Inequality (2.2), the arrival is sent to (one of) the shortest queue(s) instead of the longest queue.

In turn, to get a stochastic upper bound for SQ(d), we redirect the transitions in the following way:
1. When a departure causes the violation of Inequality (2.2), the departure may not occur.

2. When an arrival causes the violation of Inequality (2.2), the arrival is accompanied by the addition of one extra job at each of the shortest queues.

Note that adding extra arrivals in the SQ(d) upper bound model causes that we can not assume single arrivals here.

For both models, we define $X_k(t)$ to be the number of jobs in server $k$, $k = 1, 2, \ldots, N$, at time $t$, $t \geq 0$. Recall that we ordered the states such that the state space is the set

$$\mathcal{M} = \{m: m = (m_1, m_2, \ldots, m_N)\},$$

where $m_1$ denotes the largest number of jobs at the $N$ servers, $m_2$ denotes the second largest number of jobs, and so on, such that $m_N$ denotes the smallest number of jobs. By including the condition as described in Inequality (2.2), we can further describe the state space by

$$S = \{m = (m_1, \ldots, m_N) | m_j \text{ non-negative integer for } j = 1, 2, \ldots, N, \quad m_1 \geq m_2 \geq \ldots \geq m_N, \quad m_1 - m_N \leq T\}.$$

Then $X_t = (X_1(t), X_2(t), \ldots, X_N(t))$ is a Markov process as the arrivals and service times are Markovian. The main goal we want to achieve is to determine the limiting probabilities

$$\pi m = \lim_{t \to \infty} P(X(t) = m), \quad m \in S,$$

when they exist.

Before we are able to determine the limiting probabilities, we now focus on the SQ(d) lower bound model and start with the relation between the transition probabilities and we define $t_l$ as the time just before the $l$th arrival. We consider $X_l = (X_1(t_l), X_2(t_l), \ldots, X_N(t_l)); l = 1, 2, \ldots$, which is, like $X_t = (X_1(t), X_2(t), \ldots, X_N(t))$, a Markov process. This new chain $X_l = (X_1(t_l), X_2(t_l), \ldots, X_N(t_l)); l = 1, 2, \ldots$ is imbedded at arrival epochs and therefore we call it the imbedded Markov chain. For the imbedded Markov chain, we are now able to express the transition probabilities by conditioning on the interarrival time $U_l$; that is

$$p_{m, m'} = \int_0^\infty P(X_{l+1} = m' | U_l = t, X_l = m) dA(t).$$

To illustrate this equation, Figure 5.1 visualizes what the conditioning on the interarrival time $U_l$ leads to.

![Figure 5.1: Idea conditioning on the interarrival times](image)
In Figure 5.1 we observe that conditioning on the interarrival time $U_l$ leads to counting the number of departures during $U_l$. Therefore, we see that the number of jobs in the system provides useful information. As a consequence, we introduce another variable that helps us to derive the relation between transition probabilities we are looking for. For a state $m$, we define $\#m$ as the total number of jobs in state $m$. The total number of jobs includes both the jobs in service and the waiting jobs. We see that $p_{m,m} = 0$ if $\#m > \#m + 1$, because we only consider single arrivals and in case there are no departures during $U_l$, the total number of customers in the system in the new state is $\#m = \#m + 1$. Last, we define the variable $k$ as the number of departures during $U_l$, i.e. $k = (\#m + 1) - \#m'$.

We are now ready to prove the relation between certain transition probabilities, which clarifies what we observe in the structure of the generator matrix $Q$.

**Theorem 2:** Let $m, m' \in S$ be two states. If $\#m' = \#m + 1$ and $m_N > 0$ or if $\#m' < \#m + 1$ and $m_N > k$, then

$$p_{m,m'} = p_{m+1,m'+1}$$

(5.1)

where $1 = (1, 1, \ldots, 1)$.

**Proof.** Note that when $\#m' < \#m + 1$ we enforce $m_N > k$ to guarantee all servers are working during $U_l$. In the worst case scenario the $k$ departures all take place at the shortest queue $m_N$ and also in this scenario all servers must be busy during $U_l$.

We define $X_l$ to be the state immediately after the arrival of the $l$th job.

We prove Theorem 2 for the two cases $\#m = \#m + 1, m_N > 0$ and $\#m' < \#m + 1, m_N > k$ separately.

We start with the case $\#m' = \#m + 1, m_N > 0$. Here

$$p_{m,m'} = \int_0^\infty P(X_{l+1} = m' | U_l = t, X_l = m) dA(t)$$

$$= \int_0^\infty P(X_{l+1} = m' | X_l = m) P(X_{l+1} = m' | U_l = t, X_l = m') dA(t)$$

$$= \int_0^\infty P(X_{l+1} = m' | X_l = m) P(\text{no job served} | U_l = t, \text{all servers busy at } t_1) dA(t)$$

$$= \int_0^\infty P(X_{l+1} = m' | X_l = m) e^{-\mu t} dA(t)$$

$$= p_{m+1,m'+1}$$

Note that $X_{l+1} = X_l + 1$ and also $X_{l+1} = X_l + 1$ as in this case there was only one arrival and no departure. Also note that we consider $\mu$ here to make the analysis more insightful. In fact, $\mu$ is still unitary.

We continue with the case $\#m < \#m + 1, m_N > k$. For this situation we define $\rightarrow m_1 \rightarrow m_2 \ldots \rightarrow m_k$ as the event that the system is in state $m_1$ after the first job is served, in state $m_2$ after the second job is served, and so on, such that the system is in state $m_k$ after the $k$th job is served. Now

$$p_{m,m'} = \int_0^\infty \sum_{m_1, \ldots, m_k \in E} P(\rightarrow m_1 \ldots \rightarrow m_k = X_{l+1} = m' | U_l = t, X_l = m) dA(t)$$

$$= \int_0^\infty \sum_{m_1, \ldots, m_k \in E} P(\rightarrow m_1 + 1 \ldots \rightarrow m_k + 1 = X_{l+1} = m' + 1 | U_l = t, X_l = m + 1) dA(t)$$

$$= p_{m+1,m'+1}$$

where $E$ denotes the event $\{\#m_1 = \#m, \#m_2 = \#m - 1, \ldots, \#m_k = \#m - k = \#m' \}$. This concludes the proof.
5.1. MODEL DESCRIPTION

Besides this relation between transition probabilities, there is another useful theorem that we will see. In this theorem we consider \( \mu \) to make the analysis more insightful, but in fact \( \mu \) is still unitary. We will use this theorem in Section 5.2 to prove the relation between the limiting probabilities of certain states.

**Theorem 3:** Let \( m, m' \in S \) be two states. If \( \# m' = \# m + 1 \) and \( m_N > 0 \), then

\[
\sum_{m : \# m = \# m + 1} p_{m,m'} = \beta_0,
\]

where

\[
\beta_0 = \int_0^\infty e^{-\mu t} dA(t). \tag{5.3}
\]

If \( \# m' < \# m + 1 \) with \( m_0 > k \), then

\[
\sum_{m : \# m = \# m + 1 - k} p_{m,m'} = \beta_k,
\]

where

\[
\beta_k = \int_0^\infty \frac{(\mu t)^k}{k!} e^{-\mu t} dA(t). \tag{5.5}
\]

**Proof.** We start again with the case \( \# m' = \# m + 1 \). From Equation (5.1) and \( \sum_m P(X'_t = m' | X_t = m) = 1 \) it follows that

\[
\sum_{m : \# m = \# m + 1} p_{m,m'} = \sum_{m : \# m = \# m + 1} \int_0^\infty P(X'_t = m' | X_t = m) e^{-\mu t} dA(t)
\]

\[
= \int_0^\infty \sum_{m : \# m = \# m + 1} P(X'_t = m' | X_t = m) e^{-\mu t} dA(t)
\]

\[
= \int_0^\infty e^{-\mu t} dA(t) = \beta_0.
\]

Now we consider the case \( \# m' < \# m + 1, m_N > k \). From Equation (5.1) we have

\[
\sum_{m : \# m = \# m + 1 - k} p_{m,m'} = \sum_{m_1, \ldots, m_k \in E} \int_0^\infty P(\rightarrow m_1 \ldots \rightarrow m_k | U_t = t, X_t = m) dA(t)
\]

\[
= \int_0^\infty \sum_{m_1, \ldots, m_k \in E} P(\rightarrow m_1 \ldots \rightarrow m_k | U_t = t, X_t = m) dA(t)
\]

\[
= \int_0^\infty P(k \text{ jobs served} | U_t = t, \text{all servers busy at } t) dA(t)
\]

\[
= \int_0^\infty \frac{(\mu t)^k}{k!} e^{-\mu t} dA(t)
\]

\[
= \beta_k.
\]
Note again, that in the worst case scenario that the $k$ departures all take place at the shortest queue $m_N$, the condition $m_N > k$ guarantees all servers are working during $U_j$. This concludes the proof.

In this section we defined the modified models and proved some first important results about the transition probabilities in case of the SQ(1) lower bound model. We can use these results to come up with an important result for the steady-state probabilities. In the next section we will consider those. We will also see that Theorem 2 will be useful to obtain more detailed results for the steady-state probabilities.

5.2 Solution of the model

The main goal of this chapter is to determine the limiting probabilities $\pi_m$. In this section we will see an important property between different limiting probabilities. Recall that in Section 5.3 we saw the transition flow diagram of the SQ(1) lower bound model with $N = 3$ servers and threshold $T = 2$ (see Figure 5.7). For convenience, it is shown again here.

![Transition flow diagram of SQ(1) lower bound model with N = 3 and T = 2](image)

Figure 5.2: Transition flow diagram of SQ(2) lower bound model with $N = 3$ and $T = 2$

The important thing we observed in this figure was that after some initial state the process repeats itself. Note that in Section 5.4 we saw similar results for the SQ(1) upper bound model. Like we already saw in the previous section, we are able to prove for the SQ(1) lower bound model that, except for the boundaries, $p_{m,m'} = p_{m+1,m'+1}$. Here we proved what we saw in Figure 5.2 considering the repeating pattern of the transition probabilities. Although we are not able to prove this for the SQ(1) upper bound model using the imbedded Markov chain, we observe from the transition flow diagram of the SQ(1) upper bound model that also here, except for the boundaries, $p_{m,m'} = p_{m+1,m'+1}$ holds.

In this section we present the relation between corresponding states in the patterns of Figure 5.2. For this, we partition the state space in such a way that the transition matrix $Q$ will obtain a regular structure. The concepts we use are the concepts of blocks and groups of states. These techniques are also used in Zhao and Grassmann [23] and Adan et al. [1], [2] for the JSQ model.

5.2.1 Partitioning of the state space into blocks of states

First we partition the state space into blocks of states. Define the first block

$$B_{S(N-1)T} = \{m \in S | m \leq (N-1)T\}.$$  (5.6)
Note that in the transition flow diagram we already positioned the states with the same number of jobs in the system under each other. By doing so, we will see this has elegant properties for the generator matrix $Q$. The first block corresponds to the boundary states. As we have seen in Section 5.1 the weak condition for Theorems 2 and 3 is that $m_N > 0$. In other words, states for which $m_N = 0$ are included in the boundary states. Because we want the states with the same number of jobs in one group, we see that all boundary states are states $m$ for which there is a state $m'$ with $\#m = \#m'$ and $m_N = 0$. The state with the most number of jobs in the system and with $m_N = 0$ is the state $(T, T, \ldots, T, 0)$. As there are $N$ servers, the total number of jobs in this state is $(N-1)T$. Therefore, Equation (5.6) corresponds to the set of boundary states.

For the rest of the state space we define the blocks

$$B_q = \{m \in S | (N-1)T + qN < \#m \leq (N-1)T + (q+1)N\}, \quad q = 0, 1, 2, \ldots (5.7)$$

Note that we expect a regular pattern as we have seen in the transition flow diagram of Figure 5.2. In Section 5.1 we proved, except for boundary states, $p_{m,m'} = p_{m+1,m'+1}$. So we expect a corresponding state of state $m$ will be state $p_{m+1}$. The difference between these two states is the number of jobs in each server, which is one. The difference in the total number of jobs in the system is therefore $N$. Therefore, we expect that every state in set $B_q$ will correspond to exactly one state in set $B_{q+1}$ for all nonnegative integers $q$. In fact, we expect that every state in set $B_q$ will correspond to exactly one state in set $B_{q+l}$ for all nonnegative integers $q$ and $l$.

Now that we have defined the state space into blocks of states, we see that indeed

$$S = B_{\infty(N-1)T} \cup (\cup_{q=0}^{\infty} B_q).$$

Before we show the generator matrix $Q$, we order the elements within the blocks according to two criteria.

1. All states $m$ will be displayed before state $m'$ if $\#m < \#m'$.

2. If $\#m = \#m'$, the states will be ordered according to the lexicographical ordering. This means that state $m$ will be displayed before state $m'$ if
   - $m_1 > m_1'$
   - $m_1 = m_1'$ and $m_2 > m_2'$
     $\vdots$
   - $m_1 = m_1', m_2 = m_2', \ldots, m_{N-2} = m_{N-2}', m_{N-1} = m_{N-1}'$ and $m_N > m_N'$.

Note that we do not need to consider $m_1 = m_1', m_2 = m_2', \ldots, m_{N-1} = m_{N-1}'$ and $m_N > m_N'$. In that case $\#m > \#m'$, so we will order these states according to criterion 1.

### 5.2.2 Number of elements in the blocks of states

For computational reasons it will be interesting to investigate the number of elements in the blocks of states. Knowing the size of the blocks of states, we are able to say what the dimensions of the submatrices of the generator matrix $Q$ are. Knowing the number of elements in the blocks of states will later be useful when we consider the complexity of the computational algorithms. We will see this in Chapter 6.

First we compute the number of elements in block $B_q$. For the number of elements in the boundary block we give a lower and an upper bound.

**Theorem 4:** For all $q = 1, 2, \ldots$, the number of states in block $B_q$ is $\binom{(N+T-1)}{T}$. 

47
For convenience, we split the proof into four parts. In part one we introduce a different block of states by fixing \(m_1\) and adding two constraints. This block of states we will call \(B_{m_1}\). We introduce block \(B_{m_1}\) so we can easily count the number of states in this block. In part two of the proof, we will prove the number of states in block \(B_{m_1}\) is \(\binom{N + T - 1}{T}\). Then we prove there also are \(\binom{N + T - 1}{T}\) states in block \(B_0\). In the final step of the proof, we prove that for every integer \(q\) the number of elements in block \(B_q\) is also equal to \(\binom{N + T - 1}{T}\).

**Proof.**

1. Fix \(m_1 = 2T\). Define \(B_{m_1}\) as the block of states for which all states \(m\) satisfy the two constraints \(m_1 \geq m_2 \geq \ldots \geq m_N\) and \(m_1 - m_N \leq T\). Note that \#\(m\) \(\geq (N - 1)T + 1\) and therefore these states are all non-boundary states. This way we will be able to link every state in \(B_{m_1}\) to exactly one corresponding (non-boundary) state in block \(B_0\). We do this in part three of this proof.

In this part of the proof we show that all states in \(B_{m_1}\) exist for the supermarket model. Observe that the state \((m_1, m_1, \ldots, m_1)\) exists for the supermarket model. From this state one can reach every other state \(m = (m_1, m_2, \ldots, m_N)\) with \(m_1 \geq m_2 \geq \ldots \geq m_N\) and \(m_1 - m_N \leq T\). One can, for example, reach this state the following way:

\[
\begin{align*}
0 & \text{ arrivals} \\
\begin{array}{l}
m_1 - m_2 & \text{ departures at server 2} \\
m_1 - m_3 & \text{ departures at server 3} \\
& \vdots \\
m_1 - m_N & \text{ departures at server } N.
\end{array}
\end{align*}
\]

So we introduced a new block of states, for which all states exist in the supermarket model. In the next step of the proof, we will see how many states are in block \(B_{m_1}\).

2. Now we prove there are \(\binom{N + 1 - T}{T}\) states in block \(B_{m_1}\). In fact, this is a well known combinatorics problem that we denote by the "painting eggs problem". In the painting eggs problem we have \(N\) eggs and \(T\) colors and we want to know in how many ways we can paint the eggs. To see in how many ways we can paint these eggs, imagine you order the colors like

\[
\begin{align*}
N_1 & \text{ eggs of color 1} \\
N_2 & \text{ eggs of color 2} \\
& \vdots \\
N_T & \text{ eggs of color } T.
\end{align*}
\]

To gain insight, we make a sequence of zero's and ones the following way: write a zero if you see an egg and write a one if you move to the next color. For example, if you have 3 red eggs, 2 yellow eggs and 6 blue eggs, the sequence of zero's and ones would be 0001001000000. In fact, in this example we have 13 digits and 2 positions for the one. In general, we have \(N\) eggs and we move \(T - 1\) times to a different color. This means the sequence of zero's and ones has length \(N + T - 1\) and we have \(T - 1\) positions for the ones. Clearly, this gives us the quantity \(\binom{N + T - 1}{T - 1}\). So the solution to the painting eggs problem is \(\binom{N + T - 1}{T - 1}\). For a more complete proof of the painting eggs problem we refer to van Lint [15].

Counting the number of states in block \(B_{m_1}\) can now be translated to the painting eggs problem with \(N - 1\) servers (eggs) and \(T + 1\) different numbers (colors). Therefore, the number of states in \(B_{m_1}\) is \(\binom{N - 1 + T + 1 - 1}{T + 1 - 1} = \binom{N + T - 1}{T - 1}\).

3. Next we want to prove that there are \(\binom{N + T - 1}{T}\) states in \(B_0\) also. We do this by showing that for every state in \(B_{m_1}\), there is exactly one corresponding state in \(B_0\). By a corresponding state we mean that for a state \(m\) in \(B_{m_1}\), the corresponding state \(m'\) in \(B_0\) has the same structure in the sense that \(m_i - m_j = m'_i - m'_j\) for all
5.2. SOLUTION OF THE MODEL

servers \(i\) and \(j\). First we prove that for every state in \(B_{m_1}\) there is a corresponding state in \(B_0\). Then we prove that this corresponding state in \(B_0\) is also unique.

Suppose state \(m\) is one of the states in block \(B_{m_1}\), i.e. \(m_1 = 2T\) is fixed and \(m_1 \geq m_2 \geq \ldots \geq m_N\) and \(m_1 - m_N \leq T\). Suppose the number of jobs in the system is equal to

\[
\# m = \sum_{i=1}^{N} m_i = qN + (N-1)T + s, \text{ with } q \in \{0, 1, \ldots\} \text{ and } s \in \{1, 2, \ldots, N\}.
\]

Note that if \(m_1 = 2T\), then \(\# m \geq (N-1)T + 1\), so \(m\) is a non-boundary state.

Recall the definition of \(B_0\).

\[
B_0 = \{m \in S | (N-1)T < \# m \leq (N-1)T + N\}
\]

Define \(m' = m - q1\). Then

\[
\# m' = \# m - qN = (N-1)T + s.
\]

So \(m' \in B_0\).

Note that

\[
m_1 - m_2 = m'_1 - m'_2 \\
m_2 - m_3 = m'_2 - m'_3 \\
\vdots \\
m_{N-1} - m_N = m'_{N-1} - m'_N,
\]

so the structure of the state is equivalent. So for every state in \(B_{m_1}\), there is a corresponding state in \(B_0\). If we prove this state is unique we are done.

Consider \(m\) and \(m' = m - q1\). We just showed that \(m' \in B_0\). Now we show there is no other corresponding state \(m''\) that is also in block \(B_0\). Observe that all states that have the same structure as state \(m\) are of the form \(m'' = m - r1\) with \(r \in \mathbb{Z}\). For positive integers \(r\) we distinguish two cases

\[
m'' = m - (q + r)1, \quad r \in \{1, 2, \ldots\}
\]

\[
m'' = m - (q - r)1, \quad r \in \{1, 2, \ldots\}.
\]

Consider \(m'' = m - (q + r)1\), where \(r \in \{1, 2, \ldots\}\). Then

\[
\# m'' = \# m - (q + r)N \\
\leq (N-1)T.
\]

So \(m'' \notin B_0\).

Now consider \(m'' = m - (q - r)1\), where \(r \in \{1, 2, \ldots\}\). Then

\[
\# m'' = \# m - (q - r)N \\
> (N-1)T + N.
\]
So $m_0 \not\in B_0$.
So $m \in B_0$ only for $r = q$, so $m' \in B_0$ is unique.
This means that every state in $B_{m_0}$ corresponds to exactly one other state in $B_0$. As there are $\binom{N+T-1}{T}$ states in $B_{m_0}$, there are also $\binom{N+T-1}{T}$ states in block $B_0$. This is what we wanted to prove in part three of the proof. The only thing left to prove now is that for any positive integer $q$ block $B_q$ also has $\binom{N+T-1}{T}$ states.

4. Last, we want to prove that for $q \in \{1,2,\ldots\}$ the block $B_q$ also has $\binom{N+T-1}{T}$ states. To prove this, we use the same technique as we used in part three of the proof. We show that every state in $B_0$ has exactly one corresponding state in $B_q$.
Recall the definition of $B_q$.

$$B_q = \{ m \in S | (N-1)T + qN < \#m \leq (N-1)T + (q+1)N \}, \quad q = 0,1,2,\ldots$$

Suppose $m \in B_0$. Then $\#m = \sum_{i=1}^{N} m_i = (N-1)T + s$, with $s \in \{1,2,\ldots,N\}$. Note again that all states sharing the same structure as state $m$ are of the form $m' = m + r1$ with $r \in \mathbb{Z}$. Repeating the same argument as in part three of the proof, we see that there is a unique $r$ for which $m' \in B_q$. Clearly, this is $r = q$.
So for all positive integers $q$ and every state $m$ in $B_0$, there is a unique corresponding state in $B_q$. So the number of states in $B_q$ is $\binom{N+T-1}{T}$, which concludes the proof.

Now we know that the number of states in block $B_q$ is $\binom{N+T-1}{T}$, we can use this result to give an upper bound for the number of boundary states, i.e. the number of states in $B_{S(N-1)T}$. Following the same reasoning as in the previous proof, we also find a lower bound for the number of boundary states.

**Theorem 5:** The number of states in block $B_{S(N-1)T}$ is at least $\binom{N+T-1}{T}$ and at most $T^{(N+T-1)}$.

**Proof.** Both for the lower bound and for the upper bound we define the following blocks of states:

- $B_{N=0} = (m_1,m_2,\ldots,m_{N-1},0)$,
- $m_1 \geq m_2 \geq \ldots \geq m_{N-1} \geq 0$,
- $m_1 \leq T$;

- $B_{N=i} = (m_1,m_2,\ldots,m_{N-1},i)$,
- $m_1 \geq m_2 \geq \ldots \geq m_{N-1} \geq i$,
- $m_1 \leq T+i, \quad i = 1,2,\ldots,T-1$.

Note that the number of states in both $B_{N=0}$ and $B_i$ is $\binom{N+T-1}{T}$ for all $i \in \{1,2,\ldots,T-1\}$. As we have seen in part two of the proof of Theorem 4, this combinatorics problem is equivalent to the painting eggs problem with $N-1$ servers (eggs) and $T+1$ different numbers (colors).

Also note that $m_N \leq T - 1$ for all boundary states as the smallest state with $m_N = T$ is $(i,T,\ldots,T)$ which has $NT$ jobs in the system and is therefore by definition not a boundary state. In fact, all states with $m_N = 0$ are boundary states. Therefore, a lower bound for the number of states in $B_{S(N-1)T}$ is

$$\#B_{S(N-1)T} \geq \binom{N+T-1}{T}.$$

Although there are boundary states with $m_N = i$, not all states with $m_N = i$ are boundary states for $1 \leq i \leq T - 1$. For example, $(T+1,T+1,\ldots,T+1,i)$ is not a boundary state. In fact, there are many non-boundary states in the sets $B_{N=i}$. However, the upper bound we give does not take the non-boundary states in the sets $B_{N=i}$ into account as it is difficult to count those states. Therefore, the upper bound we give for the number of states in $B_{S(N-1)T}$ is

$$\#B_{S(N-1)T} \leq T^{\binom{N+T-1}{T}},$$

which concludes the proof.
5.2. SOLUTION OF THE MODEL

5.2.3 The generator matrix $Q$

In this subsection we focus on a more general method to obtain numerical results for the SQ($d$) upper bound model, that is also valid for the SQ($d$) lower bound model. In the next subsection we use a more specific method to obtain numerical results for the SQ($d$) lower bound model.

Now we are ready to show how the generator matrix $Q$ looks like. It will be well structured, which is useful to use matrix-geometric techniques in order to obtain a numerically tractable method to compute various performance characteristics for both the SQ($d$) lower and upper bound models.

$$Q = \begin{pmatrix}
R_{00} & R_{01} & 0 & 0 & 0 & \ldots \\
R_{10} & A_1 & A_0 & 0 & 0 & \ldots \\
0 & A_2 & A_1 & A_0 & 0 & \ldots \\
0 & 0 & A_2 & A_1 & A_0 & \ldots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots
\end{pmatrix}$$

Here, the submatrix $R_{00}$ corresponds to the transition rates to jump from a boundary state to another boundary state. The submatrix $R_{01}$ corresponds to the transition rates to jump from a boundary state to a non-boundary state in block $B_0$. The submatrix $R_{10}$ includes the rates to jump from a non-boundary state in block $B_0$ to a boundary state. As we observe from the generator matrix $Q$, we observe for non-boundary states that $p_{m,m'} = p_{m+1,m'+1}$. Therefore, we know that for all integers $q \geq 0$, the transition rate to move from a state $m$ in block $B_0$ to another state $m'$ in block $B_0$ will be equal to the rate to move from the corresponding state $m+q1$ in block $B_q$ to the corresponding state $m'+q1$ in block $B_q$ respectively. Therefore, except for $R_{00}$, all submatrices on the main diagonal of the generator matrix $Q$ are identical. We call this matrix $A_1$.

Next, for $q \geq 1$, we consider the rates to move from a state in block $B_q$ to a state in block $B_{q-1}$. As for the matrices $A_1$, we see that the rate to move from a state $m$ in block $B_1$ to another state $m'$ in block $B_0$ will be equal to the rate to move from the corresponding state $m+q1$ in block $B_q$ to the corresponding state $m'+q1$ in block $B_{q-1}$ respectively. So, except for $R_{01}$, all submatrices on the subdiagonal on the right of the main diagonal are identical. We call this matrix $A_2$.

Following the same reasoning as we did for $A_0$, we see that, except for $R_{10}$, all submatrices on the subdiagonal on the left of the main diagonal also are identical. The name of this submatrix is $A_2$.

Because of this structure of the generator matrix $Q$, we will see how the stationary equations look like. Now the stationary equations are given by

$$(\pi_{\leq(N-1)T}, \pi_0, \pi_q)Q = 0,$$

where $\pi_{\leq(N-1)T}$ is the limiting probability of the boundary block and $\pi_q$ is the limiting probability of block $B_q$ for all $q \in \{0, 1, 2, \ldots\}$. According to blocks we can now write the balance equations for the equilibrium probabilities.

$$0 = \pi_{\leq(N-1)T}R_{00} + \pi_0R_{10} \tag{5.8}$$
$$0 = \pi_{\leq(N-1)T}R_{01} + \pi_0A_1 + \pi_1A_2 \tag{5.9}$$
$$0 = \pi_{q-1}A_0 + \pi_qA_1 + \pi_{q+1}A_2, \quad q = 1, 2, \ldots \tag{5.10}$$

The equations (5.8) and (5.9) are called the boundary equations, and the equations in (5.10) are called the queue equations.

In the further analysis of the SQ($d$) lower and upper bound model, the matrix $R$ plays an important role. We define the elements $R_{ij}$ of the matrix $R$ as the expected number of visits to state $j$ in block $B_1$, starting from state $i$ in block $B_0$. This matrix $R$ is called the rate matrix and is characterized by

$$0 = \sum_{k=0}^{\infty} R^k A_k$$
$$= A_0 + RA_1 + R^2A_2.$$
Note that $R$ is a $\binom{N+T-1}{T}$ by $\binom{N+T-1}{T}$ matrix as the number of states in both $B_0$ and $B_1$ is $\binom{N+T-1}{T}$.

In order to use matrix-geometric techniques, we observe that the generator matrix $Q$ is irreducible, since the matrices $B_0$ and $A_1$ are non-singular as their determinant is unequal to zero. Also, assuming we have a stable system, all states are positive recurrent and, as a result, the generator matrix $Q$ is positive recurrent. Therefore, we may use Theorem 1.7.1 of Neuts [20], which states that the solutions of the stationary probabilities of the SQ($d$) lower and upper bound models can be obtained by solving the balance equations

$$
(\pi \leq (N-1)T, \pi_0, \pi_1) \begin{pmatrix} R_{00} & R_{01} & 0 \\ R_{10} & A_1 & A_0 \\ 0 & A_2 & A_1 + RA_2 \end{pmatrix} = 0,
$$

with normalization constant $\pi \leq (N-1)T e + (\pi_0 + \pi_1)(I - R)^{-1} e = 1$ and $e$ is the all one vector of proper dimensions.

As mentioned at the beginning of this chapter, the stability condition for the SQ($d$) lower bound model is $\frac{A}{D} < 1$. For the SQ($d$) upper bound model, this condition is no longer sufficient for stability as redirecting transitions to less preferable states causes more work at the servers. Therefore, the balance equations as in (5.11) only have a solution if and only if (see again Theorem 1.7.1 of Neuts [20])

$$
\pi A_0 e < \pi A_2 e,
$$

where $\pi$ is given by $\pi A = 0, \pi e = 1$, where $A = A_0 + A_1 + A_2$.

Once we know what the rate matrix $R$ is, we are able to solve this system of equations to obtain the steady-state probabilities. From there, we can obtain various performance metrics, e.g. the mean waiting time of jobs. So if we are able to obtain the rate matrix $R$, we are done.

Last, we will explain how to obtain the rate matrix $R$. For this purpose, we use the technique described in Latouche and Ramaswami [12, 13]. In Latouche and Ramaswami [13] it is explained how the matrix $G$ can be computed, where $G$ is the matrix with elements $G_{i,j}$ representing the probability that starting from a state $i$ in block $B_1$ the chain eventually visits block $B_0$ and does so by visiting state $j$. Like the rate matrix $R$, the matrix $G$ is a $\binom{N+T-1}{T}$ matrix as the number of states in both $B_0$ and $B_1$ is $\binom{N+T-1}{T}$. Also, like the rate matrix $R$, the matrix $G$ can be characterized by an equation which is in case of $G$

$$
0 = A_2 + A_1 G + A_0 G^2.
$$

The matrix $G$ for a generator matrix $Q$ is than explicitly given by

$$
G = -\sum_{k=1}^{\infty} (\prod_{i=1}^{k} B_{1,i}) B_{2,k},
$$

where

$$
B_{1,1} = (-A_1)^{-1} A_0 \\
B_{2,1} = (-A_1)^{-1} A_2 \\
B_{1,i} = (I - B_{1,i-1} B_{2,i-1} - B_{2,i-1} B_{1,i-1})^{-1} B_{1,i-1}^2 \\
B_{2,i} = (I - B_{1,i-1} B_{2,i-1} - B_{2,i-1} B_{1,i-1})^{-1} B_{2,i-1}^2.
$$

Latouche and Ramaswami [13] claim that the algorithm to compute $G$ needs only few iterations $k$. We observe this holds for our purposes and for our configurations of the system, we do not need more than $k = 6$ iterations. The rate matrix $R$ can then be computed from the matrix $G$ by (see Latouche and Ramaswami [12])

$$
R = -A_0 (A_1 + A_0 G)^{-1}.
$$
Now that we have a numerical algorithm to compute the rate matrix $R$, we are able to obtain the steady-state probabilities by solving the balance equations as in (5.11) with the normalization condition. From these equilibrium probabilities of the states, we obtain a stochastic lower and upper bound on the mean waiting time for the SQ$(d)$ model. Concretely, for each state we know how many waiting jobs there are at each server, i.e., server $i$ has $\max((m_i - 1), 0)$ waiting jobs, and we can multiply this number by the equilibrium probability of the corresponding state. By doing so for all states, we can compute the jobs' average delay in a numerically tractable manner.

In this subsection we created a numerical method to generate stochastic lower and upper bounds for the mean waiting time for the SQ$(d)$ model. In the next subsection, we introduce a more detailed method to obtain the steady-state probabilities for the SQ$(d)$ lower bound model. We will show that there is a relation between the steady-state probabilities by showing that there is a constant, say $\omega$, such that for every state, except for boundary states, is related to exactly one other state in the sense that the ratio between their equilibrium probabilities is $\omega$. We also demonstrate that this constant $\omega$ is equal to $\rho^N$, where $\rho$ is the traffic intensity and $N$ is the number of servers. Using this relation, we obtain a more direct way to obtain the steady-state probabilities for the SQ$(d)$ lower bound model.

5.2.4 The generator matrix $Q$ for the SQ$(d)$ lower bound model

In the previous subsection we already obtained a numerically tractable method to compute the steady-state probabilities for the SQ$(d)$ lower bound model. However, we can simplify this method dramatically by introducing an important relation between steady-state probabilities. Concretely, we show that for non-boundary states $\pi_{q+1} = \rho^N \pi_q$, for all $q = 1, 2, \ldots$. Therefore, we first derive the following result, which is a result for an arbitrary arrival process $A(t)$ with identically and independently distributed interarrival times with mean $1/\lambda$.

**Theorem 6**: The solutions of the stationary probabilities of the SQ$(d)$ lower bound model with an arbitrary arrival process $A_t$ have a modified vector-geometric form. Specifically,

$$\pi_{q+1} = \sigma^N \pi_q, \quad q = 1, 2, \ldots$$

and $(\pi_{\leq(N-1)}^T, \pi_0, \pi_1)$ can be obtained by solving the balance equations

$$(\pi_{\leq(N-1)}^T, \pi_0, \pi_1) \begin{pmatrix} R_{00} & R_{01} & 0 \\ R_{10} & A_1 & A_0 \\ 0 & A_2 & A_1 + \sigma^N A_2 \end{pmatrix} = (\pi_{\leq(N-1)}^T, \pi_0, \pi_1).$$

Here $\sigma$ is the unique solution for $x$, inside the unit circle, of the equation

$$x = \sum_{k \geq 0} x^k \beta_k,$$\quad (5.15)

where we defined $\beta_k$ in Section 5.1. For convenience, we split the proof into three parts. First we prove that the queue equations have a vector-geometric solution. Then we prove that this vector-geometric solution is also satisfied by the boundary states. Last, we prove that the geometric parameter $\omega$ must be equal to $\sigma^N$.

Note that the balance equations as in (5.14) differ from the balance equations as in (5.11) as we are considering the imbedded Markov chain where we first considered a continuous Markov chain. Also note that this causes that the rate matrix $R$ now satisfies

$$R = \sum_{k=0}^{\infty} R^k A_k = A_0 + RA_1 + R^2 A_2.$$\quad (5.16)

**Proof**. 1. We start by stating that the queue equations as in (5.10) have a vector-geometric solution, i.e.

$$\pi_{q+1} = \omega \pi_q.$$\quad (5.16)
To show this we need the following lemma.

**Lemma 1:** The queue equations as in (5.10) have a vector-geometric solution in the form of Equation (5.16) if and only if,

\[
\det \left[ \omega I - (A_0 + \omega A_1 + \omega^2 A_2) \right] = 0
\]

**Proof:** If we replace \( \pi_q \) by \( \omega \pi_{q-1} \) and we replace \( \pi_{q+1} \) by \( \omega^2 \pi_{q-1} \) we see Equation (5.10) becomes

\[
\omega \pi_{q-1} = \pi_{q-1} A_0 + \omega \pi_{q-1} A_1 + \omega^2 \pi_{q-1} A_2. \quad (5.17)
\]

This is a non-trivial solution and therefore

\[
\det \left[ \omega I - (A_0 + \omega A_1 + \omega^2 A_2) \right] = 0.
\]

Our generator matrix \( Q \) is irreducible and therefore satisfies the condition stated in Lemma 1.2.4 in Neuts [20]. Therefore we can use this lemma, stating that the rate matrix \( R \) can not have columns that are identically zero.

From there, it follows that \( R \) has at least one positive eigenvalue \( \omega_0 \) satisfying \( 0 < \omega_0 < 1. \)

Therefore, we can state the following theorem.

**Theorem 7:** Let \( \omega \) be an eigenvalue of the rate matrix \( R. \) Then \( \omega \) is a zero of the determinant

\[
\det \left[ \omega I - (A_0 + \omega A_1 + \omega^2 A_2) \right].
\]

**Proof:**

\[
\begin{align*}
\det \left[ \omega I - (A_0 + \omega A_1 + \omega^2 A_2) \right] &= \det \left[ (\omega I - (A_0 + \omega A_1 + \omega^2 A_2) - (R - (A_0 + RA_1 + R^2 A_2)) \right] \\
&= \det \left[ (\omega I - R) - ((\omega I - R) A_1 + (\omega^2 I - R^2) A_2) \right] \\
&= \det \left[ (\omega I - R) - ((\omega I - R)(A_1 + (\omega I + R) A_2)) \right] \\
&= \det \left[ (\omega I - R)(I - (A_1 + (\omega I + R) A_2)) \right] \\
&= \det (\omega I - R) \det \left[ I - (A_1 + (\omega I + R) A_2) \right] \\
&= 0
\end{align*}
\]

The first equation holds according to Lemma 1.2.3 of Neuts [20]. The last equation follows from the assumption that \( \omega \) is an eigenvalue of the rate matrix \( R, \) i.e. \( \det (\omega I - R) = 0. \)

Lemma 1 and Theorem 7 ensure that \( \det \left[ \omega I - (A_0 + \omega A_1 + \omega^2 A_2) \right] \) has at least one zero \( \omega_0 \) satisfying \( 0 < \omega_0 < 1. \) This means there is at least one \( \omega = \omega_0 \) such that the queue equations have a vector-geometric solution in the form of that given in Equation (5.16).

2. The boundary conditions will not necessarily be satisfied by the vector geometric solution. In this part of the proof we demonstrate that the boundary conditions will also be satisfied by the vector geometric solution. We do this by showing that for all states in \( B_0 \) with an outgoing transition rate to \( B_1, \) the solution of the boundary equations is given by Equation (5.16). In particular, we prove

\[
\begin{align*}
p_{m+1} &= \omega p_m, \quad \text{for all } m \in B \\
\pi_{q+1} &= \omega \pi_q, \quad q = 1, 2, \ldots,
\end{align*}
\]
where $B$ is defined as the set of all states in $B_0$ with an outgoing rate to $B_1$. We add the above two equations to the balance equations we already have. This gives us the following new system of equations:

\[
\begin{align*}
\pi_{S(N-1)T} &= \pi_{S(N-1)T} R_{00} + \pi_0 R_{10}, \\
\pi_0 &= \pi_{S(N-1)T} R_{01} + \pi_0 A_1 + \pi_1 A_2, \\
\pi_q &= \pi_{q-1} A_0 + \pi_q A_1 + \omega \pi_q A_2, \\
\omega \pi_q &= \pi_q (A_0 + \omega A_1 + \omega^2 A_2), \\
p_{m+1} &= \omega p_m, \quad \text{for all } m \in B.
\end{align*}
\]

We want to prove that the system of equations as in (5.18)-(5.22) has a non-trivial solution for $\pi_{S(N-1)T}$, $\pi_0$ and $\pi_1$. To do so we want to prove that both Equations (5.21) and (5.22) are redundant. We start with Equations (5.21) and look at the matrix $A_0$. We see that in matrix $A_0$ there are a lot of zero's and just some nonzero entries. These nonzero entries correspond to the transition probabilities to move from a state in the set $B_q$ to a state in the set $B_{q+1}$. In particular this holds for transition probabilities to move from a state in the set $B_0$ to a state in the set $B_1$. These states in $B_0$ are exactly the states that are defined to be in the set $B$. Due to Equations (5.22) we can therefore write $\pi_q A_0$ as $\omega \pi_{q-1} A_0$. Now Equations (5.21) are just Equations (5.20) multiplied by $\omega$ and therefore Equations (5.21) are redundant.

Now we want to prove that Equations (5.22) are redundant as well. Because $Q$ is a stochastic matrix we know

\[
\begin{align*}
R_{00} e + R_{01} e &= l e \\
R_{10} e + A_1 e &= l e - A_0 e \\
(A_2 + A_1 + A_0) e &= l e,
\end{align*}
\]

where the transpose of $e$ is $(1, 1, \ldots, 1)$ with a proper size. We add all boundary equations in (5.18) and in (5.19) and multiply both sides by $e$ to form one equation

\[
(\pi_{S(N-1)T} + \pi_0) e = (\pi_{S(N-1)T} (R_{00} + R_{01}) + \pi_0 (R_{10} + A_1) + \pi_1 A_2) e \\
= \pi_{S(N-1)T} l e + \pi_0 (l - A_0) e + \pi_1 A_2 e.
\]

Hence

\[
\pi_0 A_0 e = \pi_1 A_2 e. \quad (5.23)
\]

Adding all equations corresponding to block $B_1$ results in

\[
\pi_1 e = \pi_0 A_0 e + \pi_1 A_1 e + \omega \pi_1 A_2 e. \quad (5.24)
\]

Combining Equations (5.23) and (5.24) yields

\[
\pi_1 A_2 e = \pi_1 e - \pi_1 A_1 e - \omega \pi_1 A_2 e.
\]

Therefore

\[
\omega \pi_1 (A_2) e = \pi_1 (l - A_1 - A_2) e \\
= \pi_1 A_0 e. \quad (5.25)
\]

Combining Equations (5.23) and (5.25) results in
\[(p_1 - \omega p_0)A_0e = 0.\]

Recall that by definition
\[A_0e = \sum_{m : \#m = \#m+1} p_{m,m} = \beta_0, \quad (5.26)\]
as both \(\beta_0\) and \(A_0e\) correspond to the probability there is one arrival and no departure. This leads to
\[(p_{m+1} - \omega p_m)\beta_0 = 0, \text{ for all } m \in B. \quad (5.27)\]

So \(p_{m+1} = \omega p_m\), for all \(m \in B\). Hence Equations (5.22) are redundant.

3. In the last step of the proof we prove that \(\omega\) is equal to \(\sigma^N\), where \(\sigma\) is the unique solution, inside the unit circle, of Equation (5.15). We define groups \(G_k\) of states as follows:
\[G_k = \{m \in S | \#m = k\}, \quad k = 0, 1, 2, \ldots, \quad (5.28)\]

Therefore, group \(G_k\) consists of all states \(m\), that contain the same number \(k\) of jobs in the system. Add all stationary equations corresponding to groups \(G_0, G_1, \ldots, G_{T_N-1}\) to form one equation, and add all equations corresponding to group \(G_{T_N+k}\) to form one equation for each \(k = 0, 1, 2, \ldots\). Let \(p_k\) denote the sum of all stationary probabilities over group \(G_k\):
\[p_TN = \sum_{k=0}^{\infty} p_{T_N+k}(1 - \beta_0 - \beta_1 - \ldots - \beta_k)\]
\[p_{TN+q} = \sum_{k=0}^{\infty} p_{TN+q+k-1}\beta_k, \quad q = 1, 2, \ldots, \quad (5.29)\]

These are the balance equations for the equilibrium probabilities at arrival epochs in a GI/M/1 queueing system. Therefore
\[p_{q+TN+1} = \sigma p_{q+TN}, \quad q = 0, 1, 2, \ldots, \quad (5.29)\]

On the other hand, it follows from
\[\pi_{q+1} = \omega \pi_q, \quad q = 1, 2, \ldots\]

that
\[p_{q+(T+1)N} = \omega p_{q+TN} \quad (5.30)\]
at least for \(q = 1, 2, \ldots\). In fact, it is also valid for \(q = 0\) due to Equations (5.22). For our purpose here, it is enough to know that there is a \(q_0\) such that Equation (5.30) is valid for all \(q \geq q_0\). Combining Equations (5.29) and (5.30) gives that
\[\sigma^N p_{q+TN} = \omega p_{q+TN}\]
for all \(q = q_0, q_0 + 1, \ldots\). The only possibility is that \(\omega = \sigma^N\). This completes the proof of Theorem 6.

\[\Box\]
5.2. SOLUTION OF THE MODEL

As our arrivals are assumed to be Markovian, we actually have a stronger result for the SQ(d) lower bound model than we stated here. Because of the Markovian arrivals we are able to compute the solution \( \sigma \) for \( x \). As we will see next, this solution \( \sigma \) is equal to the traffic intensity \( \rho \). So not only did we prove there is a relation between corresponding states in blocks; we also know what this relation is. Also, we do not need to consider the imbedded Markov chain, but we can directly work with the continuous time Markov chain \( X_t \). We are now ready to state the following theorem.

**Theorem 8:** The solution of the stationary probabilities of the SQ(d) lower bound model has a modified vector-geometric form. Specifically,

\[
\pi_{q+1} = \rho^N \pi_q, \quad q = 1, 2, \ldots
\]  

(5.31)

and \((\pi_{(N-1)}T, \pi_0, \pi_1)\) can be obtained by solving the following equations:

\[
(\pi_{(N-1)}T, \pi_0, \pi_1) \begin{pmatrix} R_{00} & R_{01} & 0 \\ R_{10} & A_1 & A_0 \\ 0 & A_2 & A_1 + \rho^N A_2 \end{pmatrix} = 0.
\]  

(5.32)

Here \( \rho \) is the traffic intensity of the system.

**Proof.** We begin the proof by stating that the first part of this proof is similar to the complete proof of Theorem 6. The only thing we will do in this proof is compute the unique solution \( \sigma \) for \( x \). Recall that we defined \( \sigma \) to be the unique solution for \( x \), inside the unit circle, of Equation (5.15). For completeness, we repeat this equation here.

\[
x = \sum_{k \geq 0} x^k \beta_k.
\]

We start by showing what the \( \beta_k \)'s are for our Markovian arrivals. To make the analysis more insightful, we consider \( \mu \) in our derivation. However, \( \mu \) is still unitary. Once we know what the \( \beta_k \)'s are, we are able to compute the solution of Equation (5.15).

\[
\beta_k = \int_0^\infty (\mu t)^k e^{-\mu t} dA(t)
\]

\[
= \int_0^\infty (\mu t)^k e^{-\mu t} \lambda e^{-\lambda t} dt
\]

\[
= \lambda \int_0^\infty (\mu t)^k e^{-(\lambda + \mu) t} dt
\]

By means of induction and partial integration we can show that

\[
\beta_k = \frac{\lambda}{\mu} \frac{\mu^{k+1}}{(\lambda + \mu)^{k+1}}
\]  

(5.33)

To start the induction we consider \( k = 0 \). For \( k = 0 \) we have

\[
\beta_0 = \lambda \int_0^\infty e^{-(\lambda + \mu) t} dt
\]

\[
= -\frac{\lambda}{\lambda + \mu} e^{-(\lambda + \mu) t} \bigg|_{t=0} \bigg|_{t=\infty}
\]

\[
= \frac{\lambda}{\lambda + \mu}
\]

So for \( k = 0 \) Equation (5.33) holds. Now we assume that Equation (5.33) holds for \( k \in \mathbb{N} \) and we prove that Equation (5.33) also holds for \( k + 1 \). We see that
\[ \beta_{k+1} = \lambda \int_0^\infty \frac{(\mu t)^{k+1}}{(k+1)!} e^{-(\lambda+\mu)t} dt \]
\[ = \lambda \mu^{k+1} - \frac{\mu}{\lambda+\mu} \int_0^\infty \frac{t^{k+1}}{k!} e^{-(\lambda+\mu)t} dt \]
\[ = \frac{\mu}{\lambda+\mu} \lambda \int_0^\infty \frac{(\mu t)^k}{k!} e^{-(\lambda+\mu)t} dt \]
\[ = \frac{\mu}{\lambda+\mu} \left( \frac{\mu}{\lambda+\mu} \right)^{k+1} \]
\[ = \frac{\lambda}{\mu (\lambda+\mu)^{k+2}}. \] (5.34)

Equation (5.34) follows by the induction hypothesis. We see that for \( k + 1 \) Equation (5.33) holds. Hence, by using induction, we have proven that \( \beta_k = \frac{\lambda}{\mu (\lambda+\mu)^{k+1}} \).

Next we want to solve Equation (5.15). For convenience, we first express \( \beta_k \) in terms of the traffic intensity \( \rho \). We see
\[ \beta_k = \frac{\lambda}{\mu (\lambda+\mu)^{k+1}} \]
\[ = \left( \frac{\mu}{\lambda+\mu} \right)^{k+1} \]
\[ = \rho \left( \frac{1}{\rho+1} \right)^{k+1} \]
\[ = \rho \left( \frac{1}{\rho+1} \right)^{k+1} \]
\[ = \rho \frac{1}{\rho+1}. \]

Now we are ready to solve (5.15) for Markovian arrivals. It follows that
\[ x = \sum_{k \geq 0} x^k \beta_k \]
\[ = \sum_{k \geq 0} x^k \rho \frac{1}{(\rho+1)^{k+1}} \]
\[ = \sum_{k \geq 0} \left( \frac{x}{1+\rho} \right)^k \rho \frac{1}{(1+\rho)^{k+1}} \]
\[ = \frac{\rho}{1+\rho} \sum_{k \geq 0} \left( \frac{x}{1+\rho} \right)^k \]
\[ = \frac{\rho}{1+\rho} \frac{1}{1+\rho} \frac{1}{1+\rho - x}, |x| < 1 + \rho \]
\[ = \rho \frac{1}{1+\rho - x}. \]

So
5.3  SQ(2) LOWER BOUND MODEL WITH \( N = 3 \) AND \( T = 2 \)

To gain insight, in this section we demonstrate the SQ(2) lower bound model with \( N = 3 \) servers and threshold \( T = 2 \). The set of states can now be described by

\[
S = \{ \mathbf{m} = (m_1, m_2, m_3) \mid m_j \text{ non-negative integer integer for } j = 1, 2, 3, \]
\[
m_1 \geq m_2 \geq m_3, \]
\[
m_1 - m_3 \leq 2 \}.
\]

Although both the arrival and departure process are state-dependent, it is useful to consider them. Suppose we do not consider redirected transitions and we assume all servers have a different number of jobs. Then the transition rates are as follows:

- \( \lambda(\mathbf{m}, \mathbf{m} + \mathbf{e}_i) = (i - 1)\lambda \), with \( i = 2, 3 \);
- \( \mu(\mathbf{m}, \mathbf{m} - \mathbf{e}_i) = \mu \), with \( i = 1, 2, 3 \).

Redirecting transitions and equal queue lengths cause that the transition probabilities are very state-dependent. However, the transitions and, as a result, the transition flow diagram, still can easily be obtained. To illustrate this, in Figure 5.3 we show the transition flow diagram of the SQ(2) lower bound model with \( N = 3 \) servers and threshold \( T = 2 \).

![Transition flow diagram of SQ(2) lower bound model with \( N = 3 \) and \( T = 2 \)](figure)

Figure 5.3: Transition flow diagram of SQ(2) lower bound model with \( N = 3 \) and \( T = 2 \)

As we pointed out in Section 5.3, we observe that after some initial phase the process repeats itself. For example, look at the set of states with 5, 6, 7 jobs in the system. Compare this set of states with the set of states with 8, 9, 10 jobs in the system. We see the same pattern of transition rates between the states. When we continue looking at the

\[(1 + \rho - x)x - \rho = 0\]
\[-x^2 + (1 + \rho)x - \rho = 0\]
\[x^2 - (1 + \rho)x + \rho = 0\]
\[(x - \rho)(x - 1) = 0\]
\[x = 1 \lor x = \rho.\]

The solution \( x = 1 \) is a trivial solution. The non-trivial solution is \( x = \rho \), which concludes our proof.

\[\square\]
transition flow diagram we see this pattern is repeated over and over again. Therefore, we partition the state space according to the following blocks of states, where

\[ B_{≤4} = \{ m ∈ S | #m ≤ 4 \} \]

corresponds to the block of boundary states and

\[ B_q = \{ m ∈ S | 4 + 3q < #m ≤ 4 + 3(q + 1) \}, \quad q = 0, 1, 2, \ldots \]

corresponds to the non-boundary blocks. Concretely, we observe the first three blocks of states are

\[
\begin{align*}
B_{≤4} &= \{ (0,0,0), (1,0,0), (2,0,0), (1,1,0), (2,1,0), (1,1,1), (2,2,0), (2,1,1) \} \\
B_0 &= \{ (3,1,1), (2,2,1), (3,2,1), (2,2,2), (3,3,1), (3,2,2) \} \\
B_1 &= \{ (4,2,2), (3,3,2), (4,3,2), (3,3,3), (4,4,2), (4,3,3) \}.
\end{align*}
\]

Now we want to use these blocks to compute \((π_{≤4}, π_0, π_1)\). From Theorem 8 we know that \((π_{≤4}, π_0, π_1)\) can be obtained by solving the equations

\[
(π_{≤4}, π_0, π_1) \begin{pmatrix}
R_{00} & R_{01} & 0 \\
R_{10} & A_1 & A_0 \\
0 & A_2 & A_1 + ρ^3 A_2
\end{pmatrix} = \mathbf{0},
\]

(5.35)

where the matrices \(R_{00}, R_{01}, R_{10}, A_0, A_1\) and \(A_2\) are given by

\[
R_{00} = \begin{pmatrix}
-3\lambda & 3\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\
\mu & -(3\lambda + \mu) & 0 & 3\lambda & 0 & 0 & 0 & 0 \\
0 & \mu & -(3\lambda + \mu) & 0 & 3\lambda & 0 & 0 & 0 \\
0 & 0 & 2\mu & -(3\lambda + 2\mu) & \lambda & 2\lambda & 0 & 0 \\
0 & 0 & 0 & \mu & -(3\lambda + 2\mu) & 0 & \lambda & 2\lambda \\
0 & 0 & 0 & 0 & 3\lambda & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2\mu & 0 & -(3\lambda + 2\mu) \\
0 & 0 & 0 & 0 & 0 & 0 & 2\mu & 0 & -(3\lambda + 3\mu)
\end{pmatrix}
\]

\[
R_{01} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3\lambda & 0 & 0 \\
0 & 0 & 3\lambda & 0 & 0 & 0
\end{pmatrix}
\]

\[
R_{10} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 3\mu \\
0 & 0 & 0 & 0 & 0 & 0 & \mu & 2\mu \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
A_0 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
5.3. SQ(2) LOWER BOUND MODEL WITH $N = 3$ AND $T = 2$

\[
A_1 = \begin{pmatrix}
-(3\lambda + 3\mu) & 0 & 3\lambda & 0 & 0 & 0 \\
0 & -(3\lambda + 3\mu) & \lambda & 2\lambda & 0 & 0 \\
\mu & 2\mu & -(3\lambda + 3\mu) & 0 & \lambda & 2\lambda \\
0 & 3\mu & 0 & -(3\lambda + 3\mu) & 0 & 3\lambda \\
0 & 0 & 3\mu & 0 & -(3\lambda + 3\mu) & 0 \\
0 & 0 & 2\mu & \mu & 0 & -(3\lambda + 3\mu)
\end{pmatrix}
\]

\[
A_2 = \begin{pmatrix}
0 & 0 & 0 & 0 & 3\mu \\
0 & 0 & 0 & \mu & 2\mu \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Now we obtain the steady-state probabilities by solving the system of equations as in (5.35) and we are able to compute various performance characteristics. For example, we can compute the mean waiting time of jobs as for each state we know how many waiting jobs there are at each server, i.e., server $i$ has $\max((m_i - 1), 0)$ waiting jobs, and we can multiply this number by the equilibrium probability of the corresponding state. By doing so for all states, we can compute the jobs’ average delay. In the next chapter we demonstrate what the results are for this model for different values of $\lambda$ and compare these results with both simulations and the asymptotic result of Mitzenmacher [18]. In contrary to the asymptotic result, we will see the results for the SQ(2) lower bound model with $N = 3$ servers and threshold $T = 2$ will be remarkably close to our simulation results for the supermarket model with $N = 3$ servers.
CHAPTER 6

Numerical results

This chapter is devoted to numerical results. In order to show that for a small number of servers our results provide accurate bounds, we simulated the supermarket model in Mathematica. For comparison, we also consider the exact, but asymptotic, results obtained by Mitzenmacher [18], which we discuss here now. Mitzenmacher derives an exact expression for the expected time a job spends in the system in an asymptotic regime, i.e., when the number of servers tends to infinity. Note that the arrival rates grow in proportion with the number of servers. Now the expected jobs’ delay does not depend on the number of servers \( N \). The result is given by

\[
\sum_{i=1}^{\infty} \frac{\lambda^i d^i}{i^i}. \tag{6.1}
\]

As Equation (6.1) is obtained in an asymptotic regime, it provides accurate results for a large number of servers. However, for a much smaller system Equation (6.1) does not provide accurate results, especially not at high traffic intensity.

To show that our bounds do provide accurate results for smaller systems, a program was written in Mathematica to compute the mean waiting time of jobs using the analysis as described in Chapter 5. Also, we simulated the SQ\((d)\) model for \(10^6\) transitions (arrivals and departures), of which we ignored the first \(10^5\) when calculating the jobs’ average delay. We ignore the first \(10^5\) transitions in order to give the system time to approach equilibrium. Our simulation results for the SQ\((d)\) model agree with the simulation results for the SQ\((d)\) model obtained by Mitzenmacher [18].

The results for the SQ\((d)\) lower and upper bound models are obtained by the analysis described in Chapter 5. To test the validity of the model and the analysis, we also simulated the SQ\((d)\) lower and upper bound models using the same strategy as for the simulations of the SQ\((d)\) model, i.e., simulating the model for \(10^6\) transitions and ignoring the first \(10^5\) of them. We observed that the results for the SQ\((d)\) lower and upper bound models virtually coincide with the simulations.

The input parameters for our modified models are the number of servers \( N \), the threshold \( T \) and the number of choices \( d \). We observe that in the special case that \( d = 1 \), the mean waiting time of the SQ\((1)\) lower bound model is identical to the mean waiting time of \( N \) identical \( M/M/1 \) systems. Also, in the special case that \( d = N \), the mean waiting time in the SQ\((N)\) lower and upper bound models are identical to the mean waiting time in the JSQ lower and upper bound models as described in Adan et al. [1], respectively. We now focus on the SQ\((2)\) lower and upper bound models and first consider the effect of the threshold \( T \).
In Figure 6.1 (a-c) we show the average delay as a function of the utilization for five different curves representing the SQ(2) upper bound model with $T = 2$, the SQ(2) upper bound model with $T = 3$, the simulation results, the SQ(2) lower bound model with $T = 3$ and the SQ(2) lower bound model with $T = 2$. The first observation is that there is a tradeoff between the accuracy of the upper bounds and the computational complexity. Indeed, (a), (b) and (c) indicate that the upper bounds are quite loose for letting $T = 2$, and become significantly tighter for letting $T = 3$. However, the numerical complexity increases exponential in $T$ because the sizes of the (non-)boundary blocks in the generator matrix $Q$ are exponential in $T$. As a related remark, different values of $T$ change the stability condition for the SQ($d$) upper bound (see Section 5.2). The second observation is that the lower bounds are accurate for all three values of $N$, i.e., 3, 6, and 10. However, we also observe that the accuracy of the bounds decreases as the number of servers $N$ grows, especially for $T = 2$.

Now we want to investigate the performance of our model in comparison with the asymptotic result of Mitzenmacher. Therefore, we look in more detail at the effect of the number of servers in Figure 6.2.

In Figure 6.2 (a-i) we again show the average delay as a function of the traffic intensity; this time for four different curves representing the SQ(2) upper bound model, the simulation results, the SQ(2) lower bound model and the asymptotic result of Equation (6.1). We observe that the asymptotic result obtains inaccurate results for a small number of servers, especially at high service utilizations. Also we see that from $N = 6$ servers on, we need a threshold value of $T = 3$ to be more accurate than the asymptotic result. From $N = 9$ servers on, the threshold value $T = 3$ does not suffice anymore either to be more accurate than the asymptotic result. For more than $N = 9$ servers we could use a threshold value of $T = 4$, but the computational complexity of the system does not allow us to do this. Also, for $N = 12$ servers the asymptotic result seems to become more accurate.

Finally, we want to investigate what the effect of the number of choices $d$ is on the performance of our bounds. The results are visualized in Figure 6.3.

From Figure 6.3 (a-i) we observe that the SQ(3) upper bound model performs much better than the SQ(2) upper bound model. As a related remark, the stability condition for the SQ(3) upper bound model remains satisfied longer than in case of the SQ(2) upper bound model. Also, the SQ(3) lower bound model performs much better compared to the SQ(2) lower bound model; for $N = 12$ servers the SQ(3) lower bound model still performs well and it outperforms the asymptotic result of Equation (6.1). In contrast, the asymptotic result seems to be less accurate for $d = 3$ choices. For $N = 12$ servers, the asymptotic result is not yet accurate. Although it is worth investigating for which number of servers $N$ the SQ(3) lower bound model outperforms the asymptotic result of Equation (6.1), the numerical complexity of the system does not allow us to do this. As a concluding remark, we observe that our model obtains better results compared to the asymptotic results for Equation (6.1) as $d$ grows. To illustrate this, we also present the results for $d = 5$ choices in Figure 6.4.

Indeed, from Figure 6.4 we observe that the SQ(5) lower and upper bound models perform even better than the SQ(3) lower and upper bound models. Also, the asymptotic result of Mitzenmacher gets less accurate for $d = 5$ choices. We can explain this result by observing that the system gets more balanced when $d$ grows. The probability
that an arrival causes the violation of Inequality \[2.2\], which for completeness is stated here again,

\[ m_1 - m_N \leq T, \]

decreases when \( d \) grows. This results in less redirected arrivals and obviously this leads to more accurate results for the bounds.

Figure 6.2: Average delay as a function of the utilization for SQ(2)
Figure 6.3: Average delay as a function of the utilization for SQ(3)
Figure 6.4: Average delay as a function of the utilization for SQ(5)
In this chapter we give an overview of the thesis and we give some concluding remarks. Also we give recommendations for further research.

In this thesis we have considered the supermarket routing policy, where arriving jobs poll \( d \) out of \( N \) servers and join the one with the fewest jobs among these. A fundamental qualitative result is that SQ(2) yields an exponential improvement over SQ(1) in terms of delay, yet with a conceivably small overhead cost. This result is known as the ‘power-of-two’ result (see Mitzenmacher [18]). However, this result was obtained in an asymptotic regime. Despite its apparent simplicity, SQ(\( d \)) is very difficult to analyze in terms of the delay metric, even for a classical input with Poisson arrivals and exponential job sizes. What makes the problem particularly difficult to analyze, when \( d > 1 \), is that the generator matrix of an underlying \( N \)-dimensional Markov chain (representing, for instance, the number of jobs at each of the servers’ queues) has an irregular structure. Also in the particular case when \( d = N \), where the SQ(\( N \)) policy reduces to the Join the Shortest Queue routing policy, this irregular structure causes that it is very difficult to analyze the model in terms of the delay metric.

We introduced the JSQ model in Chapter 2. Lower and upper bounds were obtained for the JSQ model by redirecting transitions in such a way that the new system was less or more loaded than the original one. To gain insight, we have considered several approaches of obtaining these bounds for the JSQ model. The key idea of all of the approaches was that the redirected transitions transformed the original Markov chain with the inherent irregular structure into Markov chains with some regular structure.

In Chapter 3 we introduced the SQ(\( d \)) model and borrowed some of the ideas of the described approaches to construct the SQ(\( d \)) lower and upper bound models. For these modified models the key idea is also that we transformed the original Markov chain with the inherent irregular structure into Markov chains with some regular structure. We redirected transitions in such a way that the states of the original Markov chain of the SQ(\( d \)) lower and upper bound models were always sent to more or less preferable states, respectively. In Chapter 4 we proved that using this strategy indeed results in stochastic lower and upper bound models.

After this proof, in Chapter 5 we started to analyze the SQ(\( d \)) lower and upper bound models. For both the SQ(\( d \)) lower and upper bound models we observed that, after some initial phase, transition rates to jump from state to state did not change after adding the all one vector. For the SQ(\( d \)) lower bound model in particular, we also observed that, again after some initial phase, that every state is related to exactly one other state in the sense that the ratio between their equilibrium probabilities is \( \rho^N \). By introducing the concept of blocks of states and groups of...
states, we were able to form a new generator matrix which had a periodic structure. This structure caused that its analysis became amenable to matrix-geometric techniques (Neuts [20]).

We have carried out the matrix-analytical methods to compute stochastic lower and upper bounds on the average delay. The merit of the obtained bounds is that they hold in non-asymptotic regimes, and thus complement existing exact results obtained in asymptotic regimes. Numerical results in Chapter 6 revealed that there is an interesting tradeoff between the accuracy of the obtained upper bounds and the dimension of the computational complexity. Moreover, the lower bounds are remarkably tight, whereas the asymptotic results may be misleading in finite regimes as they significantly underestimate the ‘true’ results for small values of $N$, and especially at high utilizations.

Although it is worth investigating for which number of servers $N$ the $SQ(d)$ lower bound model outperforms the asymptotic result of Mitzenmacher [19], the numerical complexity of the system does not allow us to do this. For $d = 2$ choices we saw that already for $N = 12$ servers the asymptotic result obtained reasonable results, but for $d = 3$ and $d = 5$ choices the $SQ(d)$ lower bound model (using threshold $T = 3$) still outperformed the asymptotic result. Also it could be interesting to investigate higher values of the threshold $T$ as we only considered $T = 2, 3$. By allowing higher values of the threshold $T$, it is likely to obtain more accurate results. For our purposes though, the results for $T = 2, 3$ sufficed.
Nomenclature

\#m  Total number of jobs in state \( m \)
\( \lambda \)  Arrival rate
\( \mathbf{0} \)  All-zero vector
\( \mathbf{1} \)  All-one vector
\( \pi \)  Equilibrium probability vector
\( \mathbf{e} \)  All-one vector
\( \mathbf{e}_i \)  Unit vector containing only zero's except for the \( i \) \( th \) element that is one
\( m \)  State providing number of jobs at each of the servers
\( \mu \)  Departure rate
\( \rho \)  Traffic intensity or utilization
\( C \)  Capacity
\( c(\cdot) \)  Cost function
\( d \)  Number of polled servers in the supermarket model
JSQ  Join the shortest queue model
\( N \)  Number of servers
\( Q \)  Generator matrix
\( R \)  Rate matrix
\( SQ(d) \)  Supermarket model
\( T \)  Threshold
\( v_n(m) \)  Expected \( n \)-period costs when starting in state \( m \)


