THE IMPACT OF SEQUENCING POLICIES ON THE MEAN FLOW TIME OF SINGLE SERVER QUEUES

by

Kübra Tanınmış

B.S., Industrial Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2014

ACKNOWLEDGEMENTS

First of all, I would like to thank Assoc. Prof. Aybek Korugan for his support and patience throughout my entire study. This work would not be completed without his guidance and encouragement. I am also very grateful to Prof. Refik Güllü and Assist. Prof. Mustafa Hayri Tongarlak for taking part in my thesis committee and for their valuable suggestions.

I would like to thank my dear friend Vuslat for her support and friendship since the first years of the collage. She always helped me to make the right decisions when I felt doubtful. I want to thank Gökçe for her encouragement, the great times we spent together and especially for the calming tea breaks. I would also like to thank Gökalp for his helps when I needed and the rest of my colleagues for their friendship.

I am grateful to my friends Ece and Tuğçe for their valuable friendship since we share the same dorm room at high school. I thank them for their support throughout my study. And, I would like to thank Ahmet for cheering me up when I feel tired and for his patience while listening the difficulties I encountered.

I will never be able to find the correct words to express my gratitudes to my parents for their love and patience. I want to thank my beloved sisters Büşra and Şeyma for being there for me whenever I needed. I am sure that things would be much harder without their support.

ABSTRACT

THE IMPACT OF SEQUENCING POLICIES ON THE MEAN FLOW TIME OF SINGLE SERVER QUEUES

In systems with finite resources, scheduling may take significant time and this may affect the performance of the system. In this study, we are interested in the time value of optimizing a schedule. To this end, we consider a single server queue with an external scheduling mechanism and one by one Poisson arrivals. Processing times of the jobs are independent random variables following an exponential distribution. Jobs are sequenced by the scheduler for another independent exponential time following Shortest Processing Time (SPT) rule which is used here as a proxy for a general optimal scheduling rule. Two trigger mechanisms, arrivals and departures, to start a sequencing are offered. In addition to the trigger mechanisms, three rules are considered. The first one is freezing the server during sequencing. The second one is continuing processing previously sequenced jobs and the last one is continuing processing until the system is empty. In order to investigate the performance of the system with respect to mean flow time of jobs, the systems are modeled as Markov processes under each policy. To imitate an SPT queue, state dependent service rates approach is used. In numerical results part, the optimal sequencing policies under different system parameters are determined.

ÖZET

SIRALAMA POLİTİKALARININ TEK SUNUCULU SİSTEMLERDEKİ BEKLENEN AKIŞ SÜRESİNE ETKİSİ

Sınırlı kaynakları olan sistemlerde, çizelgeleme önemli bir zaman alabilir ve bu sistem performansını etkileyebilir. Bu çalışmada, bir çizelgeyi en iyilemenin zaman değeri ile ilgileniyoruz. Bu amaçla, işlerin Poisson sürecine göre gelmekte olduğu ve harici bir çizelgeleme mekanizması olan tek sunuculu bir kuyruk sistemini ele alıyoruz. İşlerin işlenme süreleri üstel dağılım izleyen rassal değişkenlerdir. İşler çizelgeleyici tarafından bağımsız diğer bir üstel zamanda, bu çalışmada genel bir en iyileme kuralını temsil eden en kısa işlem süresi (SPT) kuralına göre sıralanmaktadırlar. Bir sıralama başlatmak için iki tetik mekanizması, gelişler ve gidişler, önerilmiştir. Buna ek olarak üç kural ele alınmıştır. Bunların ilki sunucunun sıralamalar sırasında duraklatılmasıdır. İkincisi sıralamalar sırasında önceden sıralanmış olan işlerin işlenmesine devam edilmesidir ve sonuncusu sistem tamamen boşalana kadar işlemeye devam edilmesidir. Sistemin, işlerin beklenen akış süresi bakımından performansını inceleyebilmek için, sistem her bir kural altında birer Markov süreci olarak modellenmiştir. Bir SPT kuyruğunu taklit edebilmek için, duruma bağlı hizmet hızı yaklaşımı kullanılmıştır. Sayısal sonuçlar bölümünde, farklı sistem değişkenleri altında en iyi sıralama politikaları belirlenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

C	System capacity
FT	Mean flow time of the jobs in the system
FT_{SPT}	Mean flow time of jobs under SPT discipline
FT_{FCFS}	Mean flow time of jobs under FCFS discipline
L_{SPT}	Average queue length under SPT discipline
m_n	Service rate multiplier when there are n sequenced jobs in the queue
$\bar{m}$	Average service rate multiplier
N	Number of jobs in the system
T	Total number of states in the Markov Model
λ	Job arrival rate to the system
μ	Average processing rate of the jobs
μ_n	Service rate when there are n sequenced jobs in the queue
β	Sequencing rate of the jobs by the scheduler
ρ	Utilization of the server
τ	Sequencing time of the jobs by the scheduler

LIST OF ACRONYMS/ABBREVIATIONS

CTMC	Continuous Time Markov Chain
NPP	Non-preemptive Priority
SPT	Shortest Processing Time
SPTT	Truncated Shortest Processing Time
SRPT	Shortest Remaining Processing Time

1. INTRODUCTION

In finite resource systems jobs are scheduled in order to obtain an optimal usage of resources and/or satisfaction of customer needs. For different kinds of performance measures and system features, exact scheduling rules, approximation algorithms and heuristic methods that provide the best sequence and timing of jobs, are available. The performances of these rules, under various system parameters have been examined comprehensively in literature, for deterministic, off-line information [1].

On-line scheduling algorithms are needed when the system state changes and more information is obtained by time. The change in the information on hand affects the optimality of the prevalent decisions and so the performance of the system [2]. Hence, such systems are monitored and the decisions are updated according to the new information if it is needed, most of the time using simulation [3]. In some cases, continuous monitoring and scheduling may not be feasible, or it may be costly. To solve this problem, periodic or event driven dispatching policies may be applied [4].

In general, the researchers in the scheduling area have focused on the resulting sequences and their performance, not the application process of the rules. In fact, applying them was assumed to be an instantaneous work. However, in many complex systems with time as a delivery constraint, decision making processes seeking optimal sequences, take time. Spending time on the decision may reduce the benefit from scheduling for many reasons. First of all, the system state may change while the optimal sequence is being determined since the system is dynamic. This deteriorates the optimality of the decision. Another disadvantage is the decision time itself. The decision may cause a delay for the pending operations. At some point, the disadvantages may exceed the benefits and then scheduling efforts become worthless.

In this thesis we analyze these problems. Thus, the deterioration of an optimal decision in time is our focus. We address the question, is it worth scheduling when the optimal decision requires some positive time. The answer depends on the time spent to

obtain the optimal decision. It is also affected by the triggering policies that dispatch the optimization procedure. Therefore we also look at when to trigger a scheduling operation.

In order to isolate the problems of interest, we propose a simple queuing system with Poisson arrivals, exponential service and scheduling times. Here, sequencing of items following Shortest Processing Time (SPT) rule is used as a proxy for a general schedule optimization procedure. In a single server Markovian queue, SPT rule minimizes the mean flow time of items in the system and the mean queue length. The SPT rule has great attention in literature. It is known that, it also performs well for some other performance measures such as mean tardiness, mean lateness, number of late jobs etc.

Priority queues have been a center of interest in the queuing systems area. Modeling priority queues as Markov or semi-Markov processes, is a frequently used approach. Analysis of waiting lines by introducing state dependent service or inter-arrival rates to the Markov Chain model is also widespread [5]. In our study, to investigate the effect of positive sequencing time and dispatch policies on the mean flow time of single server queues we need a model that behaves like an M/M/1 queue under SPT discipline. Therefore, the problem is modeled as a Continuous Time Markov Chain with state dependent service rates.

In the following sections, first a brief survey of related studies in the literature is presented in Chapter 2. The problem and the objectives of the study are described in more detail in Chapter 3. The problem is modeled as a Continuous Time Markov Chain with state dependent service rates for alternative dispatch policies, in Chapter 4. Then, using the models created, the effect of the decision time is analyzed numerically and the results are discussed. Lastly, some possible future research ideas are presented.

2. LITERATURE SURVEY

Scheduling of tasks in systems with finite resources has been a focus of interest for researchers for decades. A great variety of studies have been carried out in order to analyze the performance of current scheduling rules and to propose new methods for more complex networks. Depending on the properties of the scheduling environments, exact analytical solutions or approximation algorithms and heuristic methods are available. In many real world scheduling problems, systems are dynamic and system state changes that occur in real time should be considered. These problems need to be treated with on-line scheduling algorithms. Most of the scheduling problems in dynamic systems are known to be NP-complete [6]. Hence, as well as exact methods, approximation algorithms have attracted significant attention in the area of research recently. One of the popular methods used in literature for the analysis of real-time scheduling problems is Markov or semi-Markov models [7].

In this thesis, an on-line scheduling environment is analyzed under different dispatch policies. The system is modeled as a Markov process using state dependent service rates. Hence the literature is examined under three topics. The first topic is on general priority sequencing. The second topic contains studies on real-time dispatching policies. The last topic presents the studies related to load dependent service rates in waiting lines and Markov process approaches.

2.1. Scheduling Rules and Priority Sequencing in Queues

Waiting lines are generally studied under FCFS discipline until Cobham's [8] paper. It is one of the first studies on priority queues. He offers an approach for the assignment of priority classes in systems with random inter-arrival and service times, and non-preemptive priorities. Waiting time is defined as the time an item spends in the system before starting being processed. In order to assign the priorities in an efficient way, Cobham obtains a closed form solution for waiting time in single server queues as a function of priority class and utilization, for any service time distribution. He

also provides the waiting time equation for multiple channel systems with exponential service times. Holley [9] focuses on the same problem as Cobham, and she simplifies the solution procedure that he provides.

Chang [10] first considers a two priority class single channel queue with different disciplines within each class. In one of the classes non-preemptive priority rule is applied and whereas preemptive resume priority rule is valid in the other one. Using Laplace transforms, he provides the distributions of queue lengths and waiting times. Later in his paper [11], the preemptive single channel case is considered for a number of priority classes. In addition to waiting times, completion times and busy period distributions are also analyzed.

Priority assignment approach used by Cobham, is applied to a machine repair line in which the job with the shortest repair time has the highest priority, by Phipps [12]. Since priority is proportional to repair time, the number of priority classes is infinite and the expected waiting time in the queue is obtained as an integral of the repair time. Therefore the resulting equation which is the waiting time in an M/M/1 queue under SPT rule, is an adaptation of Cobham's result to continuous priority classification.

Conway and Maxwell [13] deals with the optimality of shortest operation discipline under various shop floor parameters. They investigate the disadvantages of the discipline and in order to overcome them they modify the original SPT rule as two class, truncated, alternating shortest operation rules. They conclude that the rule is insensitive to errors of estimate and its disadvantages can be overcome using the modifications they propose.

Mean queuing times in a waiting line with two priority classes and bulk arrivals are presented in Hawkes' [14] paper, using Laplace transforms. The system that is analyzed has Poisson arrivals and the number of jobs in each arriving batch are independently distributed with a predefined generating function. He does not come up with an explicit solution for the distribution of queuing times. Exact steady state probabilities are computed for a two class single channel priority queue by Marks [15] under

both preemptive and non-preemptive priorty rules. Williams [16] analyzes the waiting time distributions and obtains Laplace transforms, in a multi-channel non-preemptive priority queue with two priority classes again. The service time distribution is assumed to be identical in both classes. In the study by Kella and Yechiali [17], a similar problem to Williams' is analyzed. In their case, the number of priority classes is k instead of two. They also obtain Laplace transforms of waiting times of each class.

Schrage and Miller [18] gives the Laplace transforms of waiting time and residence time of jobs in an M/G/1 preemptive priority queue. Residence time is defined as the time between first time a job's service starts and its leaving time from the system. The queue discipline is shortest remaining processing time and there is no loss of processing due to preemption. Without preemption their results are identical to results of Phipps [12]. They also compare their findings for SRPT rule with other queue disciplines such as FCFS, two class non-preemptive priority (NPP) and SPT.

Shanthikumar [19] proposes a queue discipline in which jobs with processing time less than or equal to α are sequenced under SPT and the others are served according to FCFS discipline. Initially they offer a non-linear waiting cost function. Then they compare the waiting time cost of this truncated Shortest Processing Time discipline (SPTT, α) to other priority queue disciplines and FCFS. They did not present the marginal queue length distributions as well. Buzacott and Shanthikumar [20] compare SPT, two class NPP (2C-NP) and SPTT-α, as well. They obtain mean waiting times for GI/G/c queues and conclude that SPTT-α and 2C-NP disciplines perform better than FCFS. They also state that for many instances 2C-NP performs better than the others.

There are a large number of scheduling rules for different operating conditions and performance measure criteria. Panwalkar and Iskander [1] describes and classifies around 100 problems and relevant scheduling rules for each. Blackstone *et al.* on the other hand, comparing several scheduling rules whose results are available from recent studies determines the ones with overall good performance [3]. SPT, SPTT and earliest due date (EDD) rules are a few of them.

2.2. On-line Dispatching Policies

In an on-line scheduling policy, information is obtained as jobs arrives at the system and the scheduling algorithm responds according to the new information. Sgall [6] classifies the on-line scheduling algorithms that have been studied before, according to their objective functions and to the information that is obtained on-line, basically. He compares the techniques used and the numerical results obtained.

In the study by Shanthikumar [21], three dispatch policies are examined in a single server dynamic queuing network. The scheduling operations are carried out in an external dispatch area. Jobs first arrive at this area and then released to the job shop in batches, after being scheduled. Therefore, besides the scheduling rule used, when to schedule and release the jobs is the other important decision. A cost function is determined that includes dispatching, monitoring and job waiting costs. The scheduling rule within batches is fixed as SPT. Performances of the three dispatch policies are compared with respect to mean waiting times and the cost function that has been defined before. This is the only study that considers dispatch/monitoring cost as a decision criteria.

In the studies on dynamic job shop scheduling, simulation is a very frequently used tool. Wayne and Jones [22], uses simulation to examine the problem of monitoring and updating schedules in a stochastic manufacturing environment. Similarly Kim and Kim [23] proposes a new approach that uses simulation to optimize a given performance measure in a real-time scheduling environment. The simulation takes the information of actual state of shop floor and decides the optimal dispatch policy. This optimal policy is carried out until the difference between actual performance measure and the value obtained by the simulation exceeds some limit. Then the simulation again takes the actual state information and determines a new better rule.

Later, Jeong and Kim [2] improves the simulation approach proposed by Kim and Kim [23] by extending the number of candidate dispatch policies. The results show that the time to determine a new schedule rule affects the performance significantly.

In their models they assume that decision times are small enough to be used in on-line scheduling environments.

Total flow time and total completion time in a dynamic multiprocessor system are tried to be minimized in the study by Avrahami and Azar [24]. As jobs arrived at the system, they are immediately assigned to machines to prevent migration. Using a preemptive scheduling rule, an on-line algorithm with a close performance to best off-line algorithm is presented.

Sozer [25] looks at a single machine scheduling system to investigate the effect of optimization time on the system performance with respect to earliness-tardiness cost. Using simulation, she analyzes the system performance under various dispatching policies and concludes that simple heuristic algorithms may perform better than optimal algorithms that takes time. Another study that is interested in the effect of scheduling time is by Oner [26]. She considers the time spent for sequencing in M/M/1 and M/M/∞ systems with batch arrivals. Batches arrive following Poisson distribution and the number of jobs in every batch is an independent random number. She analyzes the system performance under different dispatching policies when the scheduling time is non-negligible. Dispatching policies are based on the size of an arriving batch. The scheduling is on-line in the sense that decision of whether to sequence or not is made as the batches arrive at the system. However, once the jobs in a batch are sequenced, the order of processing for that batch cannot change later. She derives the Laplace transforms of the performance measure which is a function of flow time of jobs.

2.3. Markovian Model Approximations with State Dependent Service Rates

In queuing theory, most of the problems are not Markovian. This brings the need for new approaches to analyze many queuing systems, especially with priorities as stated by Kendall [27]. Miller [28] focuses on a single channel priority queue with two priority classes with preemptive resume discipline. Arrivals are Poisson and service time follows a general distribution. Laplace transforms of the stationary distributions

of queue length and waiting time are obtained by modeling the system as a Continuous Time Markov Chain. When the service time distribution is not exponential, imbedded Markov Chain approach is used to obtain the generating function of stationary distribution.

A single server queuing system with Poisson arrivals and state dependent service rates is investigated by Harris [5]. A random variable as a function of number of items in the system is used to generate the service time at any time. Imbedded Markov Chain approach is used to solve and compare the results of some possible state dependency circumstances. One of the other studies that makes use of this approach is by Neuts [29] where two state dependent service rate scenarios are modeled and their stationary distributions are obtained. In the paper by Herzog *et al.* [30] not only service rate but also the arrival rate is assumed to be state dependent. A recursive method is proposed for the examination of the performance of such systems, using multidimensional Markov Chains.

Another study that uses Markov Processes to analyze a priority queue is by Miller [31]. The study focuses on a single server Markovian system with two classes of customers. For both preemptive and non-preemptive cases, steady state probabilities are computed using Neuts' matrix geometric invariant vectors theory [32]. Later Miller extends the study to M/M/c non-preemptive priority queues with different arrival and service rates of each class, and computes the mean waiting times of classes [33]. For service time distributions other than exponential, Kao and Narayanan [34] models the same system as a birth-death process in order to calculate the stead state probabilities.

Yih and Thesen [7] states that some part of the on-line scheduling problems can be modeled as a semi-Markov decision process. The state space is reduced empirically since the size of the resulting state space is originally very large for matrix inversion and to be solved analytically. The reduction is done by eliminating the states that have very small probabilities observing expert schedules.

Conolly and Hadidi [35] focus on a single server queue with Poisson arrivals, and

the service rate is dependent on the time between arrivals of the item that is being served and the item that arrived before it. In other words service and inter-arrival times are correlated. They conclude that system's performance can be improved using such a service mechanism. In their subsequent study [36], service rate is dependent on the queue length instead of inter-arrival times. The average service rate is multiplied by a function of queue length and the results are compared to their previous study. It is concluded that queue length dependent service rate results in smaller number of items in the system.

Extending the state dependent service rate problem, Courtois and Georges [37] studies on a case where both arrival and service rates are state dependent. Another difference in their problem is that, the service time of a customer is dependent on the queue length at the time of its service start, rather than the current queue length. Imbedded Markov Chain approach is used to compute the waiting time distribution.

Cidon *et al.* [38] are also interested in state dependency but in such a way that, service times are independent exponential random variables and inter-arrival times are proportional to service time of predecessor job. Laplaca transforms of waiting times are obtained and the system is compared to independent service and inter-arrival time systems.

So far, state dependent service rates have been used to improve the system performance. Robinson and Hendricks [39] uses load dependent service rates, to mimic the behavior of an M/M/1/SPT queue. They first simulate an M/M/1/SPT system to obtain the steady state probabilities. Then in a FCFS system they assign the server state dependent service rates in order to obtain a close steady state probability distribution to the one that is obtained by simulation. They increase the service rate when there is a long queue and decrease when it is shorter. Finally they compare the performance of their approximation with original SPT system for various open queuing networks.

Even though a wide range of scheduling rules and solution methods are examined in the literature, the concept of scheduling time has not been introduced except the

studies by Oner [26] and Sozer [25]. [26] considers the sequencing of the jobs in a batch. Since all jobs in a batch arrive at the system at once, the scheduling operation is not completely on-line. In [25], the effect of scheduling time is analyzed using simulation. In some other papers the question of when to sequence the items is asked, however the sequencing work is assumed to be done instantaneously. In this thesis, the idea behind the load dependent service rate approach used by Robinson and Hendricks [39] is used to model an M/M/1/SPT queue as a Markov process. Due to the difficulties mentioned in the study by Yih and Thesen [7], the performances of a limited number of on-line dispatching policies in terms of mean flow time in the system are examined, assuming that scheduling is a time consuming operation.

3. OBJECTIVES AND PROBLEM DEFINITION

3.1. Objectives

The first aim of this thesis is to investigate the impact of positive decision time for optimal sequence on the mean flow time in a waiting line. When decision making is a time consuming process the question, when should we start optimizing, arises. Some trigger mechanisms should be decided to start a sequencing. Then another objective of the study becomes to analyze the effect of trigger points to start an optimization on mean flow time, in the existence of positive decision time. In order to investigate these issues, a single server Markovian queuing system is considered. The mean flow time in the system is determined as the performance measure. Since SPT rule is known to minimize this performance measure [1], a side objective is to model a waiting line that behaves like an M/M/1/SPT system, using state dependent service rates approach.

3.2. Problem Definition

In order to analyze the relationship of scheduling policies to the time cost of decision making in finite resource systems, a single server system is considered. There is a scheduler in the system that sequences the jobs following SPT rule. In our study, SPT is used as a proxy for the optimal sequencing policy. In the absence of decision time, it minimizes the mean flow time of jobs in the system and the analytical solution for SPT flow time is obtained by Phipps [12]. However, when positive sequencing time is introduced, an analytical solution is not available.

To apply the SPT rule in the existence of positive decision time, it must be decided when to start to sequence the jobs in the queue. In other words, a trigger point must be specified. The trigger point choice may affect the system performance. We propose two alternatives: arrivals and departures. In order to keep the queue of the server in the optimal order all the time, we may prefer to order the jobs upon every arrival. Alternatively we may prefer to order the queue upon every departure

from the system, in other words just before starting the service of the next job. We are interested in the performances of these two rules when sequencing takes time.

To apply these two rules, some other specifications are necessary. The first one is whether to stop the server during sequencing operations or not. If the server is freezed while the scheduler is sequencing the jobs in the queue, in arrival triggered sequencing policy, every arriving job starts a sequencing operation and the job in the server waits until sequencing completion to continue being served. In departure triggered sequencing policy on the other hand, a service completion starts a sequencing operation, the server becomes idle and it is held idle until sequencing is completed. The decision time is added to waiting times of all jobs in the queue. In both cases, if a job arrives during sequencing, the scheduler stops its current sequencing and starts reordering the jobs including the new arrival.

If the server is not stopped during sequencing, while the scheduler sequences the job in the queue, the server continues its processing operations. Under this policy system state may change during sequencing, due to service completions. The server continues to serve the jobs in the queue which changes the queue length information. In case of a departure from the system, scheduler realizes the change immediately as it realizes arrivals, and starts solving the updated problem.

The decision of not freezing the server also leads to another decision. The processor can continue processing until there are no jobs in the system, including the ones that are not placed in their optimal order yet. Alternatively, it only serves the previously sequenced jobs and waits for the scheduler to finish its work. If the server does not wait for the scheduler and starts processing the jobs that are not sequenced yet, it turns into a random order processing for those jobs and the queue discipline becomes FCFS until scheduler completes its thinking.

All sequencing policies that have been mentioned in this chapter, can be described using two sets of rules. Set 1 includes the decision epochs while Set 2 includes the rules

related to the server activity during sequencing.

$$
\begin{aligned}
&\underline{Rules} \\
Set1 \quad &= \quad \{\text{Sequence upon arrivals (A), Sequence upon departures (D)}\} \\
Set2 \quad &= \quad \{\text{Freeze the server during sequencing (F),} \\
&\qquad \text{Continue processing the previously sequenced jobs (S),} \\
&\qquad \text{Continue processing until the system becomes idle (I)}\} \\
&\underline{Policies} \\
Set1 \times Set2 \quad &= \quad \{\text{AF, AS, AI, DF, DS, DI }\}
\end{aligned}
\tag{3.1}
$$

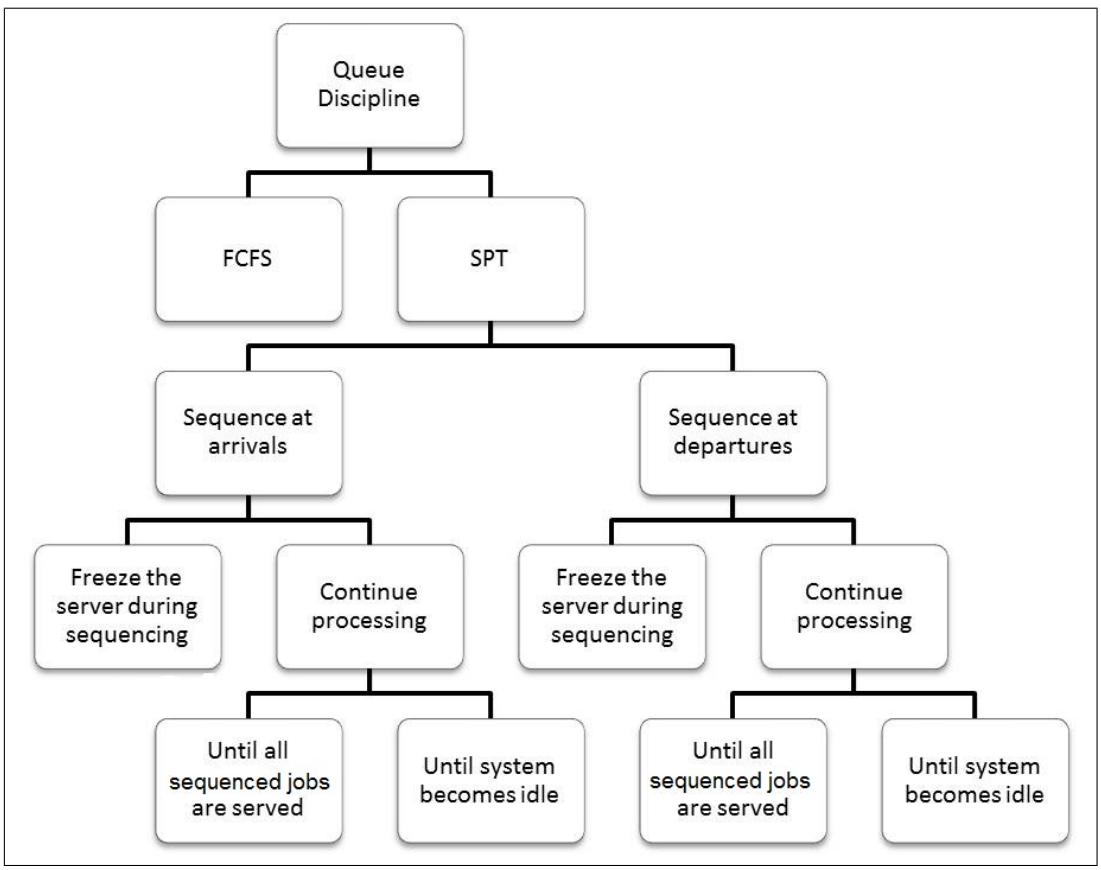


Figure 3.1. The Sequencing Policies.

As a test bed for the comparison of policies in Figure 3.1, a single server queuing system with random processing and inter-arrival times and random, load independent sequencing times, is modeled as a Markov process, since there is not an analytical solution of mean flow time for positive sequencing time. In order to emulate the SPT discipline, load dependent service rates are developed. Then the model is modified for all policies and the numerical results are obtained to discuss the performances of the sequencing policies under positive decision making time.

4. MODEL

In this study, the main purpose is to analyze the effect of positive scheduling time on the mean flow time of single server queues, under various scheduling policies. Shortest Processing Time rule is determined as a proxy for the optimal scheduling rule. The system is modeled as a Markov Process. In order to approximate the SPT discipline, state dependent service rates are used. In this chapter, first the steps followed to obtain an empirical service rate function for state dependent rates, are presented. Then, the sequencing policies mentioned in Chapter 3 are described in more detail and the transition rate matrices that are used to compute the steady state probabilities are created for all policies.

4.1. Base Model and Service Rate Function Approximation

Consider a single server queue receiving jobs following a Poisson process with rate λ, one by one. Jobs are processed by the server, again one by one for a random processing time distributed independently following an exponential distribution with rate μ. We are interested in the effect of sequencing these jobs according to the Shortest Processing Time (SPT) discipline by an external scheduling mechanism, on mean flow time, when positive sequencing time τ is introduced. τ is a random time following an independent exponential distribution with rate β.

In order to emulate the SPT queue with a Markovian Model, we use load dependent service rates approach. This approach has been used earlier by Robinson and Hendricks [39]. They use simulation to imitate the SPT discipline. This part of our study is different from theirs in the sense that, it is not desired to obtain the exact same stationary distribution as an M/M/1/SPT queue. It is desired to obtain a model that behaves similar to an M/M/1/SPT queue with respect to mean flow time in the system. SPT rule is defined in the literature as choosing the job with the smallest processing time as the next job to serve without consuming any time to identify and pick that job. Therefore to mimic the rule, a test model is needed in which the se-

quencing time is zero or negligible and the order of processing is always the same as an SPT queue. After a test model is decided satisfying these conditions, the service rate function can be determined.

Under the freeze the server during sequencing rule, the sequence of processing is optimal for sure since the server always waits for sequencing completion before choosing the next job to serve. This rule satisfies the same order of processing as an SPT queue condition. Then, the first condition is satisfied by setting the sequencing rate β to infinity. Therefore, assigning a very large number to β under the F rule results in an appropriate test bed. Freezing the server for zero time units is equivalent to not freezing in fact. This fact makes the AF and the DF policies identical. Hence, in the first phase of our study, all other policies are suspended and the system is modeled under the "Sequence at departures and freeze the server during sequencing (DF)" policy to obtain a reasonable load dependent service rate function.

In the model that is proposed, three types of transitions are possible: arrivals, service completions and sequencing completions. Arrival rate is λ as in the original problem. Sequencing completions occur with a very large rate due to the reasons discussed above. Service completion rate is determined empirically as a function of the original service rate μ and the queue length, following the reasoning that is described in the following parts.

One alternative for service rate approximation is to use the distribution of the minimum of n exponential random numbers. It is known that the minimum of n i.i.d exponential variables with rate μ is again exponential and its rate parameter is $n \times \mu$. In such a system the service rate is $n\mu$, where n is the number of jobs in the system. This system is the equivalent of an M/M/∞ system, which is the best scenario. Even though SPT decreases the mean flow time, it cannot perform as good as an infinite server system. First of all, at any time if there is a job in the server, it may not be the smallest of all jobs in the system due to arrivals that occur after its service start. Assuming that all jobs are sequenced and the one that is being processed has the smallest processing time among all, causes overestimating the service rate. For this

reason, the information of the number of jobs in the queue should be recorded in two parts.

In order to model the system, we can define a three dimensional Markov process as in Equation 4.1 where $X_1(t)$ is number of jobs arrived after the service start time of the job in the server at time t; $X_2(t)$ is the number of sequenced jobs in the queue at the service start time of the job in the server at time t; $X_3(t)$ is the server mode (idle, processing, or sequencing). The server mode presents the joint state of the server and the scheduler. The number of jobs in the queue at time t is $X_1(t) + X_2(t)$. The job in the server has the smallest processing time among the jobs that are sequenced at its service start time. Therefore, its service rate can be approximated as the rate of the smallest of $X_2(t) + 1$ independent exponential random numbers. As jobs arrive at at the system during a service operation, $X_1(t)$ increases but not $X_2(t)$, so the service rate is kept as $(X_2(t) + 1)\mu$.

$$X(t) = \{X_1(t), X_2(t), X_3(t)\} \tag{4.1}$$

A closed form solution for the stationary distribution of the queue length is not available. Thus, a finite state space is needed to be able to solve the balance equations. However the mean flow time formula is available for an M/M/1/SPT queue with infinite capacity. Therefore, to emulate an infinite capacity queue, we set the queue capacity large. Adding this capacity restriction turns the M/M/∞ type approximations to M/M/s/K type in which the system capacity K equals to the number of servers s. In our model, the queue capacity is symbolized by C and condition 4.2 specifies the limits of the resulting state space.

$$k + n \leq C \tag{4.2}$$

Having a finite state space eliminates the need for checking the stability condition of the system. For $C < \infty$, Statement 4.3 is trivial since the process has an underlying

finite, irreducible Markov Chain. Thus, at steady state, let $X(t) = x$ where $x = (k, n, s)$ determines the state of the system.

$$P\{X(t) = X\} \xrightarrow{t\to\infty} P(X) \tag{4.3}$$

The model in which we define the service rate at state $X(t)$ as $(X_2(t) + 1)\mu$, underestimates the mean flow time significantly (Figure A.1). The reason is that, for the minimum of n exponential random numbers property, we need to have n independent and identically distributed random numbers. In SPT rule, the job with smallest processing time in the sample is selected to be served. After eliminating this job from the queue, the average processing time of the jobs in the queue increases. For this reason, if there are n jobs in the system when a job starts being served, this is not an i.i.d. sample of size n with average processing time $1/\mu$. The average processing time of the jobs is larger than $1/\mu$. Hence, the job in the server is the first of n jobs with processing times following a different distribution than the initial exponential distribution with rate parameter μ.

We need a service rate function such that the model's mean flow time is between M/M/1 and M/M/∞ systems' mean flow times. From Phipps' average waiting time equation, we know the average number of jobs in a single server queue. Equation 4.4 represents the average number of jobs in an M/M/1/SPT system. The service rate function can be generated using this average number, L_{SPT}.

$$L_{\mathrm{SPT}} = \frac{\lambda^2}{\mu}\int_0^\infty \frac{\mathrm{e}^{-\mu t}\,\mathrm{d}t}{\{1 - (\frac{\lambda}{\mu})[1 - \mathrm{e}^{-\mu t}(1 + \mu t)]\}^2} \tag{4.4}$$

It is known that, even though SPT decreases the mean flow time in the system, it does not change the utilization of the server. So, $\rho_{\mathrm{SPT}} = \rho_{\mathrm{FCFS}}$. We need to propose such a service rate function that, the average service rate becomes equal to μ as in the original system. Equation 4.5 states the necessary condition in order to get the same

utilization value as in an M/M/1/FCFS system, where m_n is the multiplier of service rate μ for state n [39].

$$\bar{m} = \frac{\sum_{n=1}^{\infty} p_n m_n}{\sum_{n=1}^{\infty} p_n} = 1 \tag{4.5}$$

Steady state probabilities in an M/M/1/SPT queue have been observed by a simulation study. Results are provided in Figures A.2 through A.5. By comparing the results to FCFS case, it is concluded that SPT rule increases the probability of small states while it decreases the probability of larger ones. In order to keep the mean service rate as μ, this observation must be considered. The departure rates from larger states must be larger than μ to decrease their long run probabilities, and from smaller states must be smaller than μ to satisfy Equation 4.5.

In the literature, fastening the server when there are many jobs in the queue is a frequently used approach to improve the system performance. Here, additionally we need break even number of jobs to decrease the service rate below μ when there are less jobs than it. We know the average number of jobs in an M/M/1/SPT system. We may try to fasten the server when there are more jobs than it, and slow down when there are less jobs, so the average number of jobs gets close to the theoretical value. Then, L_{SPT} becomes a reasonable break even point. Let's call this candidate model *Model* 1 with the following service rate function where μ_n is the service rate at state $(k, n, 1)$.

$$\mu_n = (\frac{n+1}{L_{\text{SPT}}})\mu \tag{4.6}$$

Model 1 underestimates the mean flow time for small utilizations and overestimates for larger ones as in Figure 4.1. In an M/M/1 queue under FCFS discipline, the cumulative steady state probability reaches 1 fast. In other words, in probability distribution small states have the largest shares. This is more distinct in smaller utilizations. For small utilizations L_{SPT} is very small and the ratio $\frac{n+1}{L_{\text{SPT}}}$ is larger than

1 for even smallest n values which have significant probability. Therefore $\bar{\mu}$ >1. For highly utilized systems L_{SPT} is larger and $\frac{n+1}{L_{\text{SPT}}}$ is smaller than 1 for small n, decreasing $\bar{\mu}$ below 1. This is the deficiency of *Model* 1. This situation is demonstrated in Figure A.6.

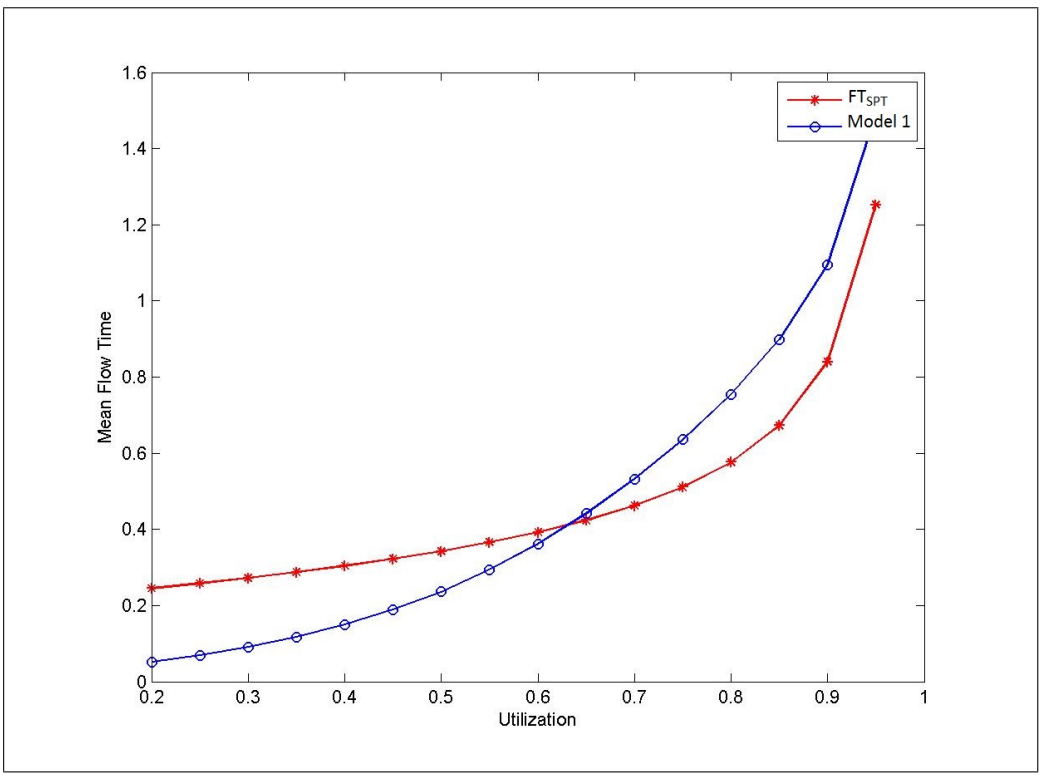


Figure 4.1. SPT vs. Model 1.

The effect of the service rate multiplier needs to be diminished especially for small utilizations, since $\bar{m}$ is farther from 1 in small utilizations. Using an exponent smaller than 1 to the $\frac{n+1}{L_{\text{SPT}}}$ term may help. Exponent should be smaller for low utilizations, which implies that it should be a function of ρ. Using directly ρ as the exponent did not decreased $\bar{m}$ enough for small utilizations. To decrease more we tried ρ^2 and finally ρ^3. Last one improved both $\bar{m}$ and mean flow time for low utilizations. New service rate as a function of n and ρ in this model (*Model* 2) is as follows:

$$\mu_n = (\frac{n+1}{L_{\text{SPT}}})^{\rho^3}\mu \tag{4.7}$$

From Figure 4.2 it seems that our model overestimates the mean flow time for high utilizations. This indicates that, the service rates need to be increased. Figure A.7 also supports this idea, $\bar{m} < 1$ for high utilizations. For both criteria, the service rate of high utilizations needs to be increased. An additional part to the service rate function without disturbing low utilizations is needed, because increasing the service rate will make the results worse for low utilizations. For this reason this additional part must be a function of ρ again.

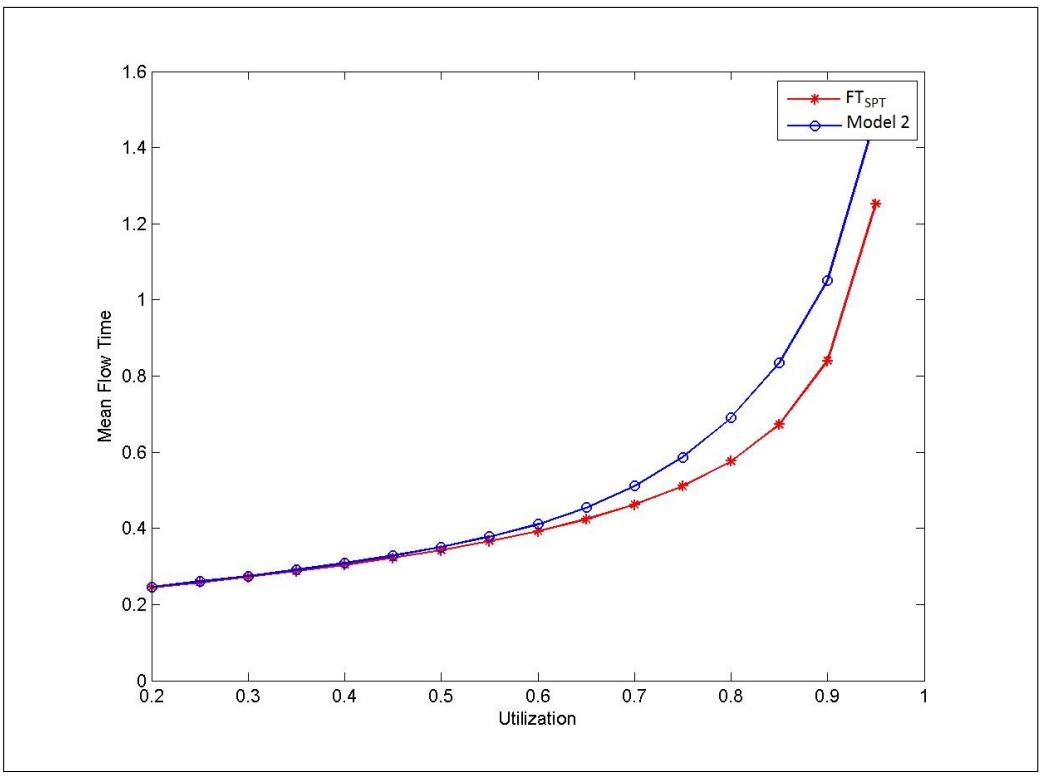


Figure 4.2. SPT vs. Model 2.

Adding a term to the numerator changes the break even point of the multiplier and decreases break even n. m_n starts to be larger than 1 for smaller n's. Adding ρ and ρ^2 decreases the break even n too much which results in $\bar{m}$ >1 for low utilizations which is not desired. Adding ρ^3 to numerator seems to help to solve this problem ($Model$ 3). Equation 4.8 presents the service rate function in this model. The resulting mean flow

time is shown in Figure 4.3.

$$\mu_n = (\frac{n+1+\rho^3}{L_{\text{SPT}}})^{\rho^3}\mu \tag{4.8}$$

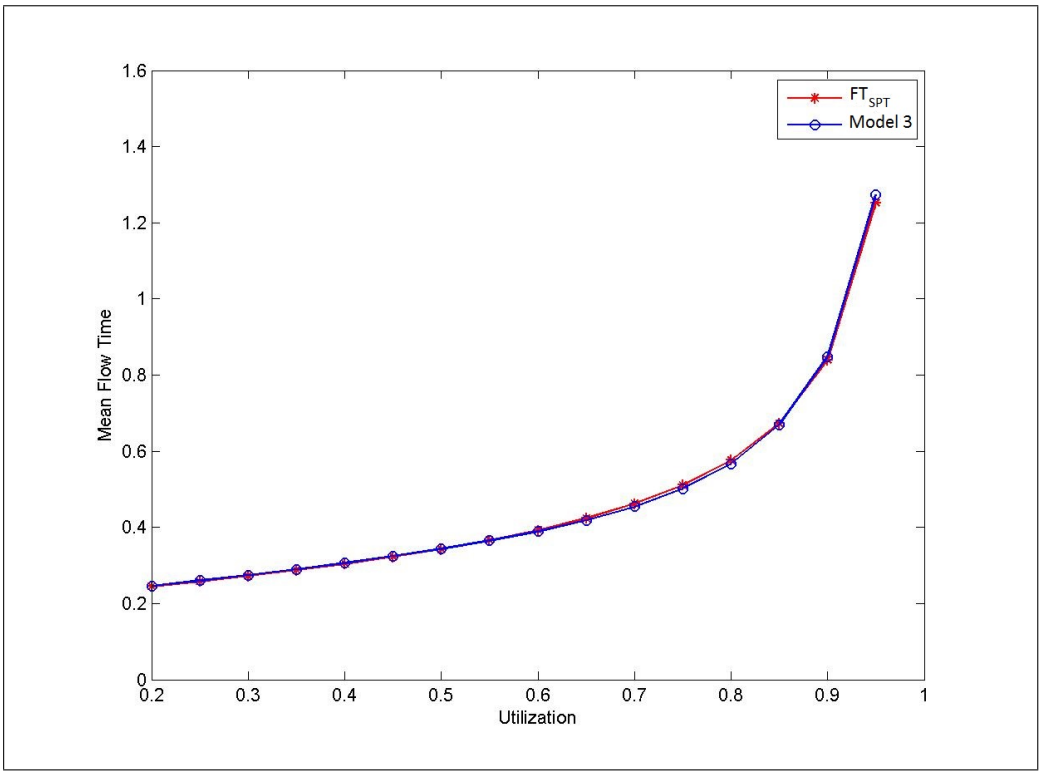


Figure 4.3. SPT vs. Model 3.

This last service rate function which is obtained through an empirical approach, imitates an M/M/1/SPT queue well with respect to mean flow time. The percent difference between the original system's and proposed model's mean flow times are presented in Figure 4.4. They are in $[-0.02, +0.02]$ interval which indicates a good estimate. Mean service rate multiplier is around 1 as desired (Figure A.8), even though it is not as good as mean flow time estimate. We model the system under the sequencing policies mentioned before, using this service rate function.

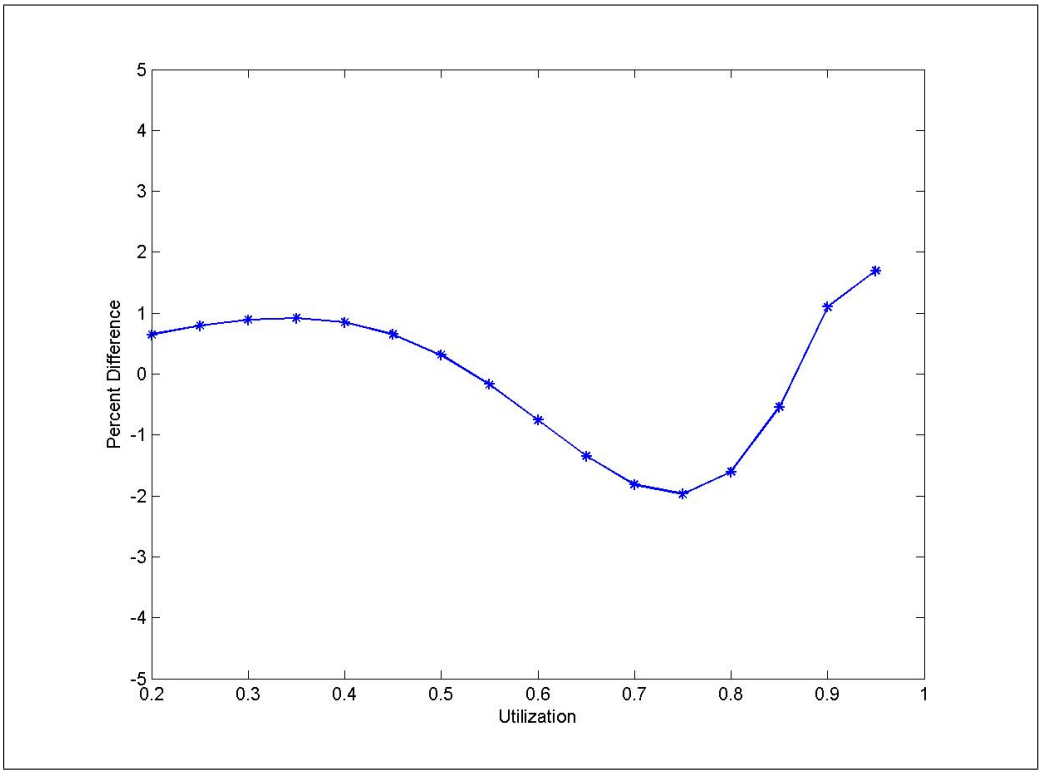


Figure 4.4. Percent Difference Between Mean Flow Time of *Model* 3 and SPT.

In the following sections, the arrival triggered policies and the departure triggered policies are described in detail. The main difference between the two policies is that, in the A policies when a job arrives at the system the scheduler starts sequencing all jobs in the queue. Thus, there is a probability that the server is not idle at the beginning of a sequencing operation. In the D policy, when the server completes the processing of a job, the scheduler starts sequencing all jobs in the queue. The server is always idle at the moment that a sequencing starts. Although there are some more variations between the Set 2 rules of the policies A and D, this main difference is valid in all policies.

4.2. Arrival Triggered Sequencing

4.2.1. Freeze the Server During Sequencing (AF)

For both decision epochs that we consider, the first sequencing rule is to freeze the server during sequencing. There are some common properties of this rule for both A and D policies. The first one is that, the server cannot continue processing a job while the scheduler is sequencing the jobs in the queue. In other words service and sequencing operations can not be performed in parallel. Secondly, during a sequencing operation, the system state can only change through arrivals. The remaining properties of the policy can be described according to the trigger point decision.

Under this policy, every arrival starts a sequencing operation unless the queue is empty. An arriving job may find the system in three basic states. In the first one both the server and the scheduler are idle, in other words the system is empty. If a job finds the system empty, its service starts immediately. In the second one, the server is busy and the scheduler is idle. If a job arrives while the server is busy, the server stops and the scheduler starts sequencing the queue. In the last state, the server is freezed and the scheduler is busy. In this state, there is a job in the server, but the server does not serve it until the sequencing is completed. When it is completed, the server resumes its previous operation. If a job arrives while there is an ongoing sequencing operation, the scheduler stops that operation and restarts sequencing the jobs in the queue including the new arrival. Here, since the sequencing time is exponentially distributed, the expected time for sequencing completion is still $1/\beta$ due to memoryless property of exponential distribution. It is expected that sequencing times are very small compared to inter-arrival times. However in systems with long sequencing times, interruption of a sequencing due to new arrivals has a higher possibility and must not be ignored.

The system state variables of this policy is defined in Table 4.1, where C symbolizes the queue capacity. The state of the queue is described with two variables in order to keep the service rate information of the job in the server, as it is mentioned in the previous section. Note that the service rate μ_n is defined as in the Equation 4.8.

The service rate is a function of n, it should stay same during a service. Hence, the jobs arrived after the service start of the job in the server is presented by k, even after the sequencing is completed. Upon service completion, all sequenced jobs is presented by n and the next job to serve is the shortest of n jobs. The transitions between states occur according to the rules in Equation Set 4.9.

Table 4.1. Definitions of State Variables for the AF Policy.

k	:	number of jobs arrived after the service start time of the job that is currently being processed (sequenced if $s = p$, being sequenced if $s = r$)	$\in \{0, 1, .., C\}$
n	:	number of jobs in the queue at the service start time of the job that is in the server currently	$\in \{0, 1, .., C - k\}$
s	:	server mode (idle, processing, sequencing)	$\in \{0, p, r\}$

Lower Boundary Equations

$$
\begin{aligned}
P(0,0,0)\lambda &= P(0,0,p)\mu_0 \\
P(0,0,p)(\lambda+\mu_0) &= P(0,1,p)\mu_1 + P(1,0,p)\mu_0 + P(0,0,0)\lambda \\
P(1,0,p)(\lambda+\mu_0) &= P(0,0,p)\lambda \\
P(0,1,p)(\lambda+\mu_1) &= P(0,2,p)\mu_2
\end{aligned}
$$

Internal Equations

$$
\begin{aligned}
P(0,n,p)(\lambda+\mu_n) &= P(0,n+1,p)\mu_{n+1} \\
&\quad + \sum_{k=1}^{C} p(k,n-k+1,p)\mu_{n-k+1} && n \geq 2 \\
P(k,n,p)(\lambda+\mu_n) &= P(k,n,r)\beta && k \geq 1, n \geq 1 \\
P(k,n,r)(\lambda+\beta) &= P(k-1,n,p)\lambda + P(k-1,n,r)\lambda && k \geq 1, k+n \geq 2
\end{aligned}
$$

Upper Boundary Equations

$$
\begin{aligned}
P(C-n,n,p)\mu_n &= P(C-n,n,r)\beta && n \geq 0 \\
P(C-n,n,r)\beta &= P(C-n-1,n,p)\lambda + P(C-n-1,n,r)\lambda && n \geq 1
\end{aligned}
\tag{4.9}
$$

Transient States: $(0, C, p)$ is transient because it cannot be reached from any state. $(0, n, p)$ can be reached from $(0, n+1, p)$ and $(k, n-k+1, p)$ in which the capacity is exceeded for $n = C$. $(1, 0, r)$ is transient becasue (k, n, r) is reached by arrivals and an arrival at $(0, 0, p)$ takes the system to $(1, 0, p)$.

During both service and sequencing operations, there is a job in the server either being served or waiting to be served. k and n present the number of jobs in the queue. Then, for the queue capacity C, the total number of jobs in the system becomes:

$$N = \begin{cases} 0 & s = 0 \\ \text{k+n+1} & s \in \{p, r\} \end{cases} \qquad \text{k+n} \leq \text{C} \tag{4.10}$$

4.2.2. Continue Processing Previously Sequenced Jobs (AS)

In this policy, it is possible to serve the jobs and sequence the queue in parallel. An arriving job that finds the server busy and the queue occupied ($k+n \geq 1$), triggers a sequencing operation. During the sequencing, the server continues its service. In fact, regardless of the state of the scheduler, the server continues to serve the jobs in the system until there are no sequenced jobs left. Then, it stops and waits for the completion of the sequencing operation.

At any time, there are four possible scenarios. In the first one, the system is empty. In the second one, the server is busy and the scheduler is idle. In the third one, both the server and the scheduler is busy and in the last scenario, the scheduler is busy while the server is waiting idle for the sequencing completion.

Since the server continues serving as long as there are sequenced jobs in the system, we should keep track of the number of sequenced jobs. Hence, in addition to the number of jobs at the service start time of the job in the server, the number of jobs arrived after then should be recorded in two parts as sequenced and unsequenced jobs. Instead of adding one more variable, this condition is satisfied by extending the set of

variable s. Therefore the state space in the previous policy is extended and the system state is defined with still three variables as (k, n, s), which are described in Table 4.2. C symbolizes the system capacity including the one in the server. In the AF policy, n presents the number of jobs in the queue while here it also includes the one in the server. The service rate function μ_n is revised accordingly.

Table 4.2. Definitions of State Variables for the AS Policy.

k	:	number of jobs that are being sequenced	$\in \{0,1,..,C\}$
n	:	number of sequenced jobs	$\in \{0, 1, .., C-k\}$
s	:	server mode	$\in \{0, p, sp, 1, ..., C-1\}$

The server continues processing as long as the number of sequenced jobs $n > 0$ for all of s. $s = p$ means the server is processing the smallest of n jobs and there are no jobs that are being ordered. Since this is an arrival triggered scheduling policy and every arrival starts a sequencing (changing the server mode to sp), $k = 0$ for $s = p$.

The server mode $s = sp$ describes a system state in which there are k new jobs that are being inserted to the queue in the right order ($k > 1$), and the server is serving a job if $n > 0$. In $(k, 0, sp)$ states, since the number of sequenced jobs is zero, the server waits idle for the sequencing completion. If $n > 0$ all transition types are possible out of these states. If a service completion occurs at these states, the server picks the next sequenced job to serve, and the scheduler starts reordering the jobs realizing this change. If a sequencing completion takes place before a service completion, all jobs in the queue become sequenced while the server is busy. Adding k to n to represent this situation, causes to lose the service rate parameter. If it is not added, then the number of sequenced jobs information is lost. This is the reason behind new integer values of variable s.

$s \in \{1, 2, ..., C-1\}$ indicates that, a sequencing operation has been completed during a service and s keeps its service rate information. As soon as the current service operation is completed, system state switches to $s = p$ or $s = sp$ depending

on k. Positive k indicates an ongoing sequencing. A service completion then, takes the system to $s = sp$. $k = 0$ means all jobs in the queue are sequenced and a service completion takes the system to $s = p$. For simplicity we refer this state set by $s = i$. All transitions are given in Equation Set 4.11.

Lower Boundary Equations

$$
\begin{aligned}
P(0,0,0)\lambda &= P(0,1,p)\mu_1 && \\
P(0,1,p)(\lambda+\mu_1) &= P(0,0,0)\lambda + P(0,2,p)\mu_2 + P(1,1,sp)\mu_1 && \\
P(1,1,sp)(\lambda+\mu_1) &= P(0,1,p)\lambda + P(1,2,sp)\mu_2 && \\
\text{Internal Equations} & && \\
P(0,n,p)(\lambda+\mu_n) &= P(0,n+1,p)\mu_{n+1} + \sum_{s=1}^{n} P(0,n+1,i)\mu_i && \\
&\quad + P(n,0,sp)\beta && n \geq 2 \\
P(k,0,sp)(\lambda+\beta) &= P(k-1,0,sp)\lambda + P(k,1,sp)\mu_1 && k \geq 2 \\
P(1,n,sp)(\lambda+\mu_n+\beta) &= P(0,n,p)\lambda + P(1,n+1,sp)\mu_{n+1} && \\
&\quad + \sum_{i=1}^{n} P(1,n+1,i)\mu_i && n \geq 2 \\
P(k,n,sp)(\lambda+\mu_n+\beta) &= P(k-1,n,sp)\lambda + P(k,n+1,sp)\mu_{n+1} && \\
&\quad + \sum_{i=1}^{n} P(k,n+1,i)\mu_i && k \geq 2, n \geq 1 \\
P(0,n,i)(\lambda+\mu_i) &= P(n-i,i,sp)\beta + \sum_{k=1}^{n-2} P(k,n-k,i)\beta && i \geq 1, n > i \\
P(k,n,i)(\lambda+\mu_i+\beta) &= P(k-1,n,i)\lambda && k \geq 1 \\
\text{Upper Boundary Equations} & && \\
P(0,C,p)\mu_C &= P(C,0,sp)\beta && \\
P(C,0,sp)(\beta) &= P(C-1,0,sp)\lambda && \\
P(C-n,n,sp)(\mu_n+\beta) &= P(C-n-1,n,sp)\lambda && n \geq 1 \\
P(C-n,n,i)\beta &= P(C-n-1,n,i)\lambda && i \geq 1, n > i
\end{aligned}
\tag{4.11}
$$

Transient States: $(1, 0, sp)$ is transient because it cannot be reached from any state. $(k, 0, sp)$ is reached from $(k-1, 0, sp)$ or $(k, 1, sp)$. $(0, 0, sp)$ does not exist and $(1, 1, sp)$ takes the system to $(0, 1, p)$. (0,2,1) is transient since there is no β out of $(1, 1, sp)$. $(k, 2, 1)$ are transient since they are only reached from (0,2,1) through arrivals.

4.2.3. Continue Processing Until System Becomes Idle (AI)

This policy is similar to the AS policy regarding that the server and the scheduler can work in parallel. Differently, the server continuous serving the jobs as long as the system is not empty. If there are not any sequenced jobs left and the scheduler is busy, the server does not stop and picks an unsequenced job to serve. Since the job is randomly selected and unsequenced, the expected processing time is $1/\mu$, not the smallest of the jobs in the queue. In such occurrences, the queue discipline turns to FCFS until the sequencing is completed, and the service completion transitions occur through rate μ. The previous model is modified such that the service rate function used is decided based on the number of ordered jobs, n. It should be noted that if there are sequenced jobs ($n > 0$), the server never serves the unsequenced ones.

Table 4.3. Definitions of State Variables for the AI Policy.

k	:	number of jobs that are being sequenced	$\in \{0,1,..,C\}$
n	:	number of sequenced jobs	$\in \{0, 1, .., C-k\}$
s	:	server mode	$\in \{0, p, sp, f, 1, ..., C-1\}$

Since the server prefers random order processing to waiting idle for a sequencing completion, a random processing operation can only start during a sequencing. In the previous model, when a sequencing operation is completed during an ongoing service, there are three numbers to record. The first one is the service rate of the job in the server, in other words the number of sequenced jobs at its service beginning. The second one is the total number of sequenced jobs in the system and the last one is the number of unsequenced jobs which arrives after the last sequencing completion. Hence, a sequencing completion takes the system to $s \in \{1, ..., C-1\}$ states. Here, if the same

rule is applied, the random processing information is lost since $(n > 0)$ in $s = i$ states. In order to represent an ongoing random processing, a new server mode $s = f$ is added to the model. Transitions to those states occur through scheduling completions during a random processing, i.e. at $(k, 0, sp)$ states as shown in Equation Set 4.12. State (k, n, f) tells that there is a job in the server with expected processing time $1/\mu$, there are $n-1$ sequenced jobs in the queue and k jobs are being sequenced. If $k = 0$ there is not an ongoing sequencing operation, but only random processing. The state variables and their definitions in Table 4.3 differ only by this new server mode from the model of the AS policy.

Lower Boundary Equations

$$
\begin{aligned}
P(0,0,0)\lambda &= P(0,1,p)\mu_1 \\
P(0,1,p)(\lambda+\mu_1) &= P(0,0,0)\lambda + P(0,2,p)\mu_2 + P(1,1,sp)\mu_1 \\
P(1,0,sp)(\lambda+\mu) &= P(2,0,sp)\mu \\
P(2,0,sp)(\lambda+\mu) &= P(1,0,sp)\lambda + P(3,0,sp)\mu + P(2,1,sp)\mu_1 \\
P(1,1,sp)(\lambda+\mu_1) &= P(0,1,p)\lambda + P(1,2,sp)\mu_2
\end{aligned}
$$

Internal Equations

$$
\begin{aligned}
P(0,n,p)(\lambda+\mu_n) &= P(0,n+1,p)\mu_{n+1} + P(0,n+1,f)\mu \\
&\quad + \sum_{i=1}^{n} P(0,n+1,i)\mu_i && n \geq 2 \\
P(k,0,sp)(\lambda+\mu+\beta) &= P(k-1,0,sp)\lambda + P(k+1,0,sp)\mu \\
&\quad + P(k,1,sp)\mu_1 && k \geq 3 \\
P(1,n,sp)(\lambda+\mu_n+\beta) &= P(0,n,p)\lambda + P(1,n+1,sp)\mu_{n+1} \\
&\quad + P(1,n+1,f)\mu + \sum_{i=1}^{n} P(1,n+1,i)\mu_i && n \geq 2 \\
P(k,n,sp)(\lambda+\mu_n+\beta) &= P(k-1,n,sp)\lambda + P(k,n+1,sp)\mu_{n+1} \\
&\quad + P(k,n+1,f)\mu + \sum_{i=1}^{n} P(k,n+1,i)\mu_i && k \geq 2, n \geq 1 \\
P(0,n,f)(\lambda+\mu) &= P(n,0,sp)\beta && n \geq 3 \\
P(k,n,f)(\lambda+\mu+\beta) &= P(k-1,n,f)\lambda && k \geq 1, n \geq 3
\end{aligned}
$$

$$
\begin{aligned}
P(0,n,i)(\lambda+\mu_i) &= P(n-i,i,sp)\beta + \sum_{k=1}^{n-2} P(k,n-k,i)\beta && i \geq 1, n > i \\
P(k,n,i)(\lambda+\mu_i+\beta) &= P(k-1,n,i)\lambda && k \geq 1 \\
&\underline{\text{Upper Boundary Equations}} \\
P(C,0,sp)(\mu+\beta) &= P(C-1,0,sp)\lambda \\
P(C-n,n,sp)(\mu_n+\beta) &= P(C-n-1,n,sp)\lambda && n \geq 1 \\
P(C-n,n,f)(\mu+\beta) &= P(C-n-1,n,f)\lambda && n \geq 3 \\
P(C-n,n,i)(\mu_i+\beta) &= P(C-n-1,n,i)\lambda && i \geq 1, n > i
\end{aligned}
\tag{4.12}
$$

Transient States: $(k,2,1)$ are transient since they are only reached from (0,2,1) through arrivals and $(0,2,1)$ cannot be reached from any state. $(k,2,f)$ is transient since it can be only reached from $(0,2,f)$ which cannot be reached from any state, since a sequencing is not carried out at $(2,0,sp)$.

4.3. Departure Triggered Sequencing

The second of Set 1 rules is to sequence the queue upon service completions. Under this policy, at the end of every processing operation, the scheduler checks whether there is a new job that arrived after the last sequencing completion or not. If there is not any new job, it implies that the sequence is still optimal and the server processes the first job in this sequence. If there are some new jobs, the queue needs to be reordered for optimal sequence. The modeling of the system under D policies is explained in detail in the following sections.

4.3.1. Freeze the Server During Sequencing (DF)

In this policy, while the scheduler is busy the server waits for the completion of sequencing, to select the next job to serve. As in the AF policy, there are three possible scenarios. The first one is the empty system. In the second one the server is busy and

the scheduler is idle, and in the last one the server is idle and the scheduler is busy. If a job arrives while the server is busy, it waits for service completion to be sequenced with all jobs in the queue. If a job arrives while the scheduler is busy, the scheduler realizes the system state change and restarts sequencing. Jobs that arrive at an empty system are immediately starts being served. The system state is again defined as (k, n, s). Descriptions of variables are given in Table 4.4. The queue capacity is C.

Table 4.4. Definitions of State Variables for the DF Policy.

k	:	number of jobs that arrived after the last sequencing completion	$\in \{0, 1, .., C\}$
n	:	number of jobs in the queue at the last service start time	$\in \{0, 1, .., C-k\}$
s	:	server mode (idle, processing, sequencing)	$\in \{0, p, r\}$

Transitions between states occur according to the rules in Equation Set 4.13. Upon a sequencing completion the number of unsequenced jobs k is added to the number of sequenced jobs n, without causing any information loss, since there is not a job in the server to keep its service rate information as in the AF policy. Hence, k always presents the number of unsequenced jobs.

Lower Boundary Equations

$$P(0,0,0)\lambda = P(0,0,p)\mu_0$$

$$P(0,0,p)(\lambda+\mu_0) = P(0,1,p)\mu_1 + P(1,0,p)\mu_0 + P(0,0,0)\lambda$$

$$P(0,1,p)(\lambda+\mu_1) = P(0,2,p)\mu_2$$

Internal Equations

$$P(0,n,p)(\lambda+\mu_n) = P(0,n+1,p)\mu_{n+1} + \sum_{k=1}^{C} P(k,n-k+1,r)\beta \qquad n \geq 2$$

$$P(k,n,p)(\lambda+\mu_n) = P(k-1,n,p)\lambda \qquad k \geq 1, n \geq 0$$

$$P(1,n,r)(\lambda+\beta) = P(1,n,p)\mu_n \qquad n \geq 1$$

$$P(k,n,r)(\lambda+\beta) = P(k,n,p)\mu_n + P(k-1,n,r)\lambda \qquad k \geq 1, k+n \geq 2$$

Upper Boundary Equations

$$\begin{aligned} P(C-n,n,p)\mu_n &= P(C-n-1,n,p)\lambda & n \geq 0 \\ P(C-n,n,r)\beta &= P(C-n-1,n,r)\lambda + P(C-n,n,p)\mu_n & n \geq 1 \end{aligned} \tag{4.13}$$

Transient States: $(0, C, p)$ is transient because it cannot be reached from any state since $(0, C+1, p)$ and $(k, C-k+1, r)$ do not exist.

In the AF policy, during a sequencing there is always a job in the server waiting for the sequencing completion to continue being served. Here, sequencing upon departures causes the server to be physically empty during sequencing, and the number of jobs in the system at any time becomes as in the Equation 4.14.

$$N = \begin{cases} 0 & s = 0 \\ \text{k+n+1} & s = p \\ \text{k+n} & s = r \end{cases} \qquad \text{k+n} \leq \text{C} \tag{4.14}$$

4.3.2. Continue Processing Previously Sequenced Jobs (DS)

Under this policy, the server is not kept idle if there are sequenced jobs in the queue. At service completion times, the queue is checked for unsequenced jobs. If there are some unsequenced jobs, the scheduler starts sequencing the as the server starts processing the smallest of sequenced jobs. In other words even there are unsequenced jobs in the system, the server does not wait for sequencing to serve the next job. If the sequencing is not completed before this next job's service, the server picks again the smallest of remaining sequenced jobs. This continues until there are not any sequenced jobs in the queue.

Modifying the model of the AS policy, the transition rate matrix is constructed

under the transition rules given in Equation Set 4.15. The system state parameters have the same definitions that are given in Table 4.2 with a small difference that k is the number of jobs arrived after the last sequencing completion. Note that since n presents the number of sequenced jobs in the system, the service rate function μ_n is revised accordingly.

Another difference from the AS policy is that sequencing is carried out in only $s = sp$ states, and not in $s \in \{1, ..., C-1\}$ $(s = i)$ states. The reason is that, transitions to $s = i$ states occur to keep the necessary informations when a sequencing operation is completed during a service, and the next sequencing starts at the following departure. A departure takes the system to $s \in \{p, sp\}$ states since the previous service rate information is not needed any more. Hence, sequencing is not carried out in $s = i$ states. As a final note, there is an exception for service completions at (k, n, sp) states. At $(1, 2, sp)$, there is one job in the server and the remaining two are being ordered. A departure decreases the number of jobs in the system to two and the number of the jobs in the queue to 1. Hence, there are no jobs to order. Instead the transition that is used is $(1, 2, sp) \xrightarrow{\mu_n} (1, 1, p)$.

Lower Boundary Equations

$$
\begin{aligned}
P(0,0,0)\lambda &= P(0,1,p)\mu_1 \\
P(0,1,p)(\lambda+\mu_1) &= P(0,0,0)\lambda + P(1,1,p)\mu_1 + P(0,2,p)\mu_2 \\
P(1,1,p)(\lambda+\mu_1) &= P(0,1,p)\lambda + P(1,2,p)\mu_2 + P(1,2,sp)\mu_2
\end{aligned}
$$

Internal Equations

$$
\begin{aligned}
P(0,n,p)(\lambda+\mu_n) &= P(0,n+1,p)\mu_{n+1} + \sum_{i=1}^{n} P(0,n+1,i)\mu_i \\
&\quad P(n,0,sp)\beta && n \geq 2 \\
P(k,n,p)(\lambda+\mu_n) &= P(k-1,n,p)\lambda && k \geq 1, n \geq 0 \\
P(k,0,sp)(\lambda+\beta) &= P(k-1,0,sp)\lambda + P(k,1,p)\mu_1 \\
&\quad + P(k,1,sp)\mu_1 && k \geq 2 \\
P(1,n,sp)(\lambda+\mu_n+\beta) &= P(1,n+1,p)\mu_{n+1} + P(1,n+1,sp)\mu_{n+1}
\end{aligned}
$$

$$
\begin{aligned}
& & & + \sum_{i=1}^{n} P(1, n+1, i)\mu_i & & n \geq 2 \\
P(k, n, sp)(\lambda + \mu_n + \beta) & = & & P(k-1, n, sp)\lambda + P(k, n+1, sp)\mu_{n+1} & & \\
& & & + P(k, n+1, p)\mu_{n+1} + \sum_{i=1}^{n} P(k, n+1, i)\mu_i & & n \geq 1, k \geq 2 \\
P(0, n, i)(\lambda + \mu_i) & = & & P(n-i, i, sp)\beta & & i \geq 1, n > i \\
P(k, n, i)(\lambda + \mu_i) & = & & P(k-1, n, i)\lambda & & k \geq 1 \\
\underline{\text{Upper Boundary Equations}} & & & & & \\
P(0, C, p)\mu_C & = & & P(C, 0, p)\beta & & \\
P(C-n, n, p)\mu_n & = & & P(C-n-1, n, p)\lambda & & n \geq 1 \\
P(C, 0, sp)\beta & = & & P(C-1, 0, sp)\lambda & & \\
P(C-n, n, sp)(\mu_n + \beta) & = & & P(C-n-1, n, sp)\lambda & & n \geq 0 \\
P(C-n, n, i)\mu_i & = & & P(C-n-1, n, i)\lambda & & i \geq 1, n > i
\end{aligned}
\tag{4.15}
$$

Transient States: $(1, 0, sp)$ is transient because it cannot be reached from any state since $(1, 1, sp)$ takes to $(0, 1, p)$ through service completion. $(1, 1, sp)$ is transient because it can only be reached from $(1, 2, 1)$ and $(k, 2, 1)$ are transient. $(1, C-1, sp)$ is transient because it cannot be reached from any state since the capacity is exceeded.

4.3.3. Continue Processing Until System Becomes Idle (DI)

This policy is an extension of the DS policy, as the AI policy is an extension of the AS policy. The server continues processing as long as the system is not empty. It never waits idle for a sequencing completion. So if there are not any sequenced jobs, it processes unsequenced jobs for a random time with expectation $1/\mu$. In order to keep the random processing information, again a new server mode $s = f$ is added to the previous model. This is the only difference in the system state variable definition of the DS policy. The balance equations are presented in Equation Set 4.16.

The service rates in (k, n, sp) states are assigned according to n. $n = 0$ indicates a random order processing, while $n > 0$ implies the processing of the smallest of n jobs. In $s = f$ state set the service rate is μ due to random order processing. Sequencing is carried out only at (k, n, sp) states. Two exceptional states are $(1, 0, sp)$ and $(2, 0, sp)$. The number of the jobs in the queue at these states are zero and one, respectively. Even though there is not an ongoing sequencing in these states, they exist to indicate random processing. Since this is a departure triggered policy, arrivals should not start a schedule. However according to balance equations of this policy given in Equation Set 4.16, an arrival takes the system from $(2, 0, sp)$ to $(3, 0, sp)$ in which sequencing is carried out. This is changed to $(2, 0, sp) \xrightarrow{\lambda} (2, 1, f)$ which tells there are two unsequenced jobs in the queue; the one in the server has an expected $1/\mu$ processing time and there is not an ongoing sequencing, as desired.

Lower Boundary Equations

$$
\begin{aligned}
P(0,0,0)\lambda &= P(0,1,p)\mu_1 + P(1,0,sp)\mu \\
P(0,1,p)(\lambda+\mu_1) &= P(0,0,0)\lambda + p(1,1p)\mu_1 + P(0,2,p)\mu_2 \\
P(1,0,sp)(\lambda+\mu) &= P(2,0,sp)\mu + P(1,1,f)\mu \\
P(2,0,sp)(\lambda+\mu) &= P(1,0,sp)\lambda + P(3,0,sp)\mu \\
&\quad + P(2,1,sp)\mu_1 + P(2,1,f)\mu \\
P(2,1,f)(\lambda+\mu) &= P(2,0,sp)\lambda
\end{aligned}
$$

Internal Equations

$$
\begin{aligned}
P(0,n,p)(\lambda+\mu_n) &= P(0,n+1,p)\mu_{n+1} + \sum_{i=1}^{n} P(0,n+1,i)\mu_i \\
&\quad P(0,n+1,f)\mu && n \geq 2 \\
P(k,n,p)(\lambda+\mu_n) &= P(k-1,n,p)\lambda && k \geq 1, n \geq 0 \\
P(k,0,sp)(\lambda+\mu+\beta) &= P(k-1,0,sp)\lambda + P(k,1,p)\mu_1 \\
&\quad + P(k,1,f)\mu + P(k,1,sp)\mu_1 \\
&\quad + P(k+1,0,sp)\mu && k \geq 3 \\
P(1,n,sp)(\lambda+\mu_n+\beta) &= P(1,n+1,p)\mu_{n+1} + P(1,n+1,sp)\mu_{n+1}
\end{aligned}
$$

$$
\begin{aligned}
& & & + \sum_{i=1}^{n} P(1, n+1, i)\mu_i & & n \geq 2 \\
P(k, n, sp)(\lambda + \mu_n + \beta) & = & & P(k-1, n, sp)\lambda + P(k, n+1, sp)\mu_{n+1} & & \\
& & & + P(k, n+1, p)\mu_{n+1} + P(k, n+1, f)\mu & & \\
& & & + \sum_{i=1}^{n} P(k, n+1, i)\mu_i & & n \geq 1, k \geq 2 \\
P(0, n, f)(\lambda + \mu) & = & & P(n, 0, sp)\beta & & n \geq 3 \\
P(k, n, f)(\lambda + \mu) & = & & P(k-1, n, f)\lambda & & k \geq 1, n \geq 1 \\
P(0, n, i)(\lambda + \mu_i) & = & & P(n-i, i, sp)\beta & & i \geq 1, n > i \\
P(k, n, i)(\lambda + \mu_i) & = & & P(k-1, n, i)\lambda & & k \geq 1 \\
\underline{\text{Upper Boundary Equations}} & & & & & \\
P(C-n, n, p)\mu_n & = & & P(C-n-1, n, p)\lambda & & n \geq 1 \\
P(C, 0, sp)\beta & = & & P(C-1, 0, sp)\lambda & & \\
P(C-n, n, sp)(\mu_n + \beta) & = & & P(C-n-1, n, sp)\lambda & & n \geq 2 \\
P(C-n, n, f)\mu & = & & P(C-n-1, n, f)\beta & & n \geq 3 \\
P(C-n, n, i)\mu_i & = & & P(C-n-1, n, i)\lambda & & i \geq 1, n > i
\end{aligned}
\tag{4.16}
$$

Transient States: $(0, C, p)$, $(1, C-1, sp)$ and $(0, C, C-1)$ are transient because they cannot be reached from any state since the capacity is exceeded. $(1, 1, f)$ is transient since (k, n, f) ,is reached from $(0, n, f)$ and $(0, 1, f)$ does not exist.

5. NUMERICAL RESULTS AND INTERPRETATIONS

We model the problem as a Markov process in the previous chapter. Since a closed form solution for the stationary distribution of the queue length is not available, we set the system capacity finite to solve the balance equations. In order to imitate an infinite capacity SPT queue whose mean flow time equation is available, we set the queue capacity large. The AF and DF policies are identical in terms of mean flow time when the sequencing rate β is very large. Adopting the DF policy arbitrarily, we have obtained an empirical service rate function.

The Markov processes of the AF and the DF policies have significantly smaller number of states than the other policies. Hence, the computations of steady state probabilities for those policies for a large system capacity is not very difficult. The size of the state space increases rapidly with the system capacity in the other policies. Consider the AS policy. For $s = p$; $n \in \{1, ..., C\}$ and $k = 0$ which yields C states. For $s = sp$; $n \in \{0, ..., C-1\}$ and $k \in \{1, ..., C-n\}$ which yields $\frac{C(C+1)}{2}$ states. Lastly for $s \in \{1, ..., C-1\}$; $n \in \{s+1, ..., C\}$ and $k \in \{0, ..., C-n\}$. This set outnumbers the first two sets. The number of states is given in Equation 5.1 and adding the empty system state, total number of states in this policy is given by Equation 5.2.

$$T_{s\in\{1,...,C-1\}} = (\frac{1.2 + ... + (C-1).C}{2}) \tag{5.1}$$

$$T = C + \frac{C(C+1)}{2} + (\frac{1.2 + ... + (C-1).C}{2}) + 1 \tag{5.2}$$

Due to the size of the state space, in our analysis we cannot consider very large capacities. The system capacity used in experiments that are discussed in this section is 40 and we compare our results with an M/M/1/40/FCFS queue's performance. The experimental results for different capacity values are presented in Appendix B.

The system is modeled as a CTMC using the state transition informations explained in Chapter 4 in detail, under all the sequencing policies that we offer. Once the transition rate matrix is obtained in MATLAB environment, the steady state probabilities are computed in GAMS solving the balance equations. Using these probabilities, the expected number of jobs in the system is calculated as in the Equation 5.3. Since the system has finite capacity, the effective arrival rate is computed using the acceptance probability of an arriving job as in the Equation 5.4

$$L = \sum_{s} \sum_{n} \sum_{k} (k+n) P(k,n,s) \tag{5.3}$$

$$\lambda_{\mathit{eff}} = \lambda.(1 - \sum_{s} \sum_{n} P(C-n,n,s)) \tag{5.4}$$

where C is the system capacity. Equation 5.5 represents the derivation of the expected flow time using Little's Law.

$$FT = \frac{L}{\lambda_{\mathit{eff}}} \tag{5.5}$$

In order to observe the effect of sequencing time τ and utilization ρ, experimental sets are designed. τ is given as a proportion of the service time of a single job, $1/\mu$. For all $\tau \in \{0, 0.1/\mu, 0.5/\mu, 1/\mu, 1.5/\mu, 2/\mu\}$ and $\rho \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95\}$, mean flow times of jobs in the system are computed using the steady state probabilities. For each policy 6×8=48 experiments and for the control set (FCFS flow times) 8 experiments were conducted resulting a total number of 296 experiments. In order to approximate the equivalent of $\beta = 1/\tau$ for $\tau = 0$, we consider a very large β value, $\beta = 100000\mu$.

The main decision is to find the best trigger point to start a sequencing operation that minimizes the FT. Hence, the A policies are compared with their corresponding D policies in terms of mean flow time. The first comparison is done under the F rule.

Table 5.1 presents the mean flow times of AF and DF policies. Since $\beta = 100000\mu$ implies zero sequencing time, AF and DF policies are indifferent in terms of mean flow time. $FT_A = FT_D = FT_{SPT}$, where FT_{SPT} is found by Equation 4.4.

Table 5.1. The F Policy Experimental Results.

Policy AF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.323	**0.403**	0.546	0.757	1.074	0.333
	0.50	0.344	0.372	**0.517**	0.802	1.268	2.020	0.400
	0.60	0.390	0.432	**0.656**	1.104	1.815	2.872	0.500
	0.70	0.454	0.513	**0.819**	1.381	2.164	3.168	0.667
	0.80	0.567	0.647	**1.035**	1.652	2.389	3.220	0.999
	0.85	0.669	0.764	1.198	**1.835**	2.548	3.312	1.321
	0.90	0.849	0.964	1.461	**2.136**	2.853	3.591	1.877
	0.95	1.274	1.429	2.062	**2.884**	3.720	4.516	2.800
Policy DF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.315	**0.355**	0.424	0.519	0.650	0.333
	0.50	0.344	0.357	**0.426**	0.551	0.740	1.024	0.400
	0.60	0.390	0.409	**0.510**	0.702	0.999	1.446	0.500
	0.70	0.454	0.480	0.614	**0.866**	1.239	1.764	0.667
	0.80	0.567	0.600	0.769	**1.069**	1.481	2.007	0.999
	0.85	0.669	0.707	0.895	1.222	**1.651**	2.177	1.321
	0.90	0.849	0.893	1.111	1.478	**1.944**	2.492	1.877
	0.95	1.274	1.334	1.628	2.111	2.701	**3.365**	2.800

As β decreases, FT increases faster in the AF policy. The reason is that in both policies, when a job completes its service it leaves an optimally ordered queue behind. Under the AF policy, this is achieved by stopping the server at every arrival. In case of more than one arrival during the service time of a job, all jobs in the system are incurred that many sequencing delays. However in the DF policy, the server is only stopped at departures if there are new jobs in the queue since the last sequencing. Even if there are more than one new job in the system, sequencing only once before

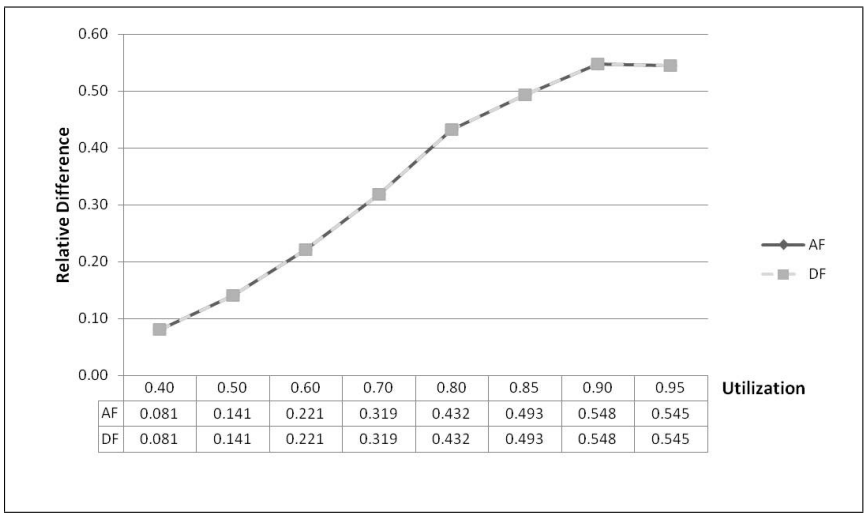
starting a new service is enough to be serving the jobs in optimal order.

This result may stem from the assumption that the sequencing time is load independent. Inserting one job in the right position in an ordered queue has the same expected duration as inserting n jobs. Therefore if n jobs arrive at the system during a service, in the AF policy the system spends an expected $n \times \tau$ time units while in the DF policy the system spends only τ time units.

The bold numbers in Table 5.1 indicate the first time $FT_{SPT} > FT_{FCFS}$ for the capacited system. Since the performance measure to minimize by the help of sequencing is the mean flow time in this study, these numbers implies that sequencing is more costly than it is beneficial. It does not make sense to sequence jobs after these points. Under both policies, this break-even β values decreases as the utilization increases. In other words, in systems with higher utilization rates, spending more time on sequencing is allowed without becoming worse than FCFS discipline. To explain this result, let's define the relative difference in mean flow time due to sequencing as in the Equation 5.6.

$$Relative\ Difference = \frac{FT_{FCFS} - FT_{SPT}}{FT_{FCFS}} \tag{5.6}$$

In the absence of decision time, while the utilization increases, the relative difference increases as it can be seen from Figure 5.1. When there are more jobs in the queue, sequencing them decreases mean flow time more. As an extreme example if there is only one job in the queue, ordering is not beneficial at all. Since we are studying finite capacity systems, after some point the difference stop increasing. The reason is that, capacity is a control mechanism for stability. The mean flow time cannot increase when the system is fully utilized, since the queue length cannot increase. Sequencing in this case decreases the queue length causing more items to enter the system, in other words increasing the effective arrival rate. The mean flow time increases as the effective arrival rate increases. This can be the explanation of the trend in Figure 5.1.



	0.40	0.50	0.60	0.70	0.80	0.85	0.90	0.95
AF	0.081	0.141	0.221	0.319	0.432	0.493	0.548	0.545
DF	0.081	0.141	0.221	0.319	0.432	0.493	0.548	0.545

Figure 5.1. Relative Difference Under AF and DF Policies for $\tau = 0$.

Figure 5.2 presents the relative difference in the mean flow time for all sequencing rates under the AF policy. Negative difference indicates that the resulting FT is larger than FT_{FCFS}. Relative difference is an increasing function of utilization, for larger sequencing rates. However, there is an interesting trend for small β. Consider $\beta = 0.5\mu$. Even though the difference is negative all the time, it first decreases then increases as the utilization increases. This can be explained by the trade-off between the number of interruptions in unit time and the number of jobs in the system. From small to medium utilizations, the sequencing does not make a significant change since the average number in the system is not large, but due to increasing arrival rates services are interrupted more frequently causing the difference to decrease. As the utilization increases more, the average number of jobs increases causing the sequencing become more effective and time spent for sequencing becomes less costly. The situation is similar for the DF policy and it is shown in Figure B.1.

Additionally, it is observed that for the DF policy the break-even β is smaller than of the AF policy for high utilizations. This means more time can be spent for sequencing under the DF policy. This can be explained by the first observation which is as β decreases, FT increases faster in the AF policy. Since in the DF policy this increase is slower, sequencing may be still beneficial for some β which is not beneficial under the AF policy. This situation is more apparent for higher utilizations because

when there are more jobs in the queue on the average, sequencing time per job decreases in the DF policy.

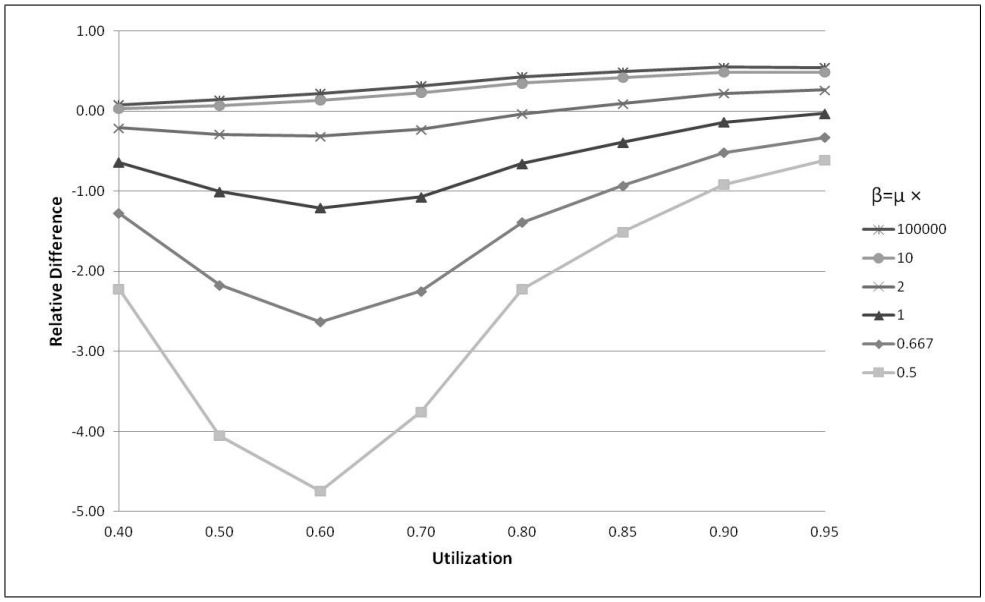


Figure 5.2. Relative Difference Under the AF Policy.

It can be concluded that, for the F policies sequencing at departures is always better than sequencing at arrivals, in terms of mean flow time in system. However under both rules the performance measure quickly reaches the break-even point since keeping the server idle in a non empty system incurs a significant time cost. Note that the break-even β is not the exact sequencing rate resulted in $FT > FT_{FCFS}$ for the first time, but the largest β giving this result in our test set.

The results of the S policies are presented in Table 5.2. The first observation is that for $\beta = 100000\mu$, $FT_{AS} = FT_{SPT}$ whereas $FT_{DS} > FT_{SPT}$. This situation stems from a substantial disadvantage of the DS policy. When a job's service is completed, the queue is not in the optimal sequence if some jobs arrived since the last sequencing. There are some jobs that have been sequenced in the last sequencing and the others have never been sequenced whose numbers are symbolized by n and k in relevant model, respectively. The S rule requires picking up the smallest job among this sequenced set. If this is not the job with smallest processing time among all jobs in the queue (sequenced and unsequenced), the sequence of processing becomes suboptimal. The

service of this job starts at the same time as the sequencing starts. As a result, even though sequencing takes zero time unit, not waiting for its completion and picking up a small but not smallest job diminishes the benefit from sequencing. The DS policy performs worse than SPT as it can be seen in $\beta = 100000\mu$ column of Table 5.2.

Table 5.2. The S Policy Experimental Results.

Policy AS		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.307	0.316	**0.337**	0.366	0.403	0.333
	0.50	0.344	0.345	0.358	0.390	**0.435**	0.491	0.400
	0.60	0.390	0.392	0.412	0.455	**0.516**	0.590	0.500
	0.70	0.454	0.459	0.487	0.542	0.617	**0.705**	0.667
	0.80	0.567	0.576	0.617	0.684	0.768	0.864	0.999
	0.85	0.669	0.680	0.731	0.806	0.893	0.989	1.321
	0.90	0.849	0.864	0.928	1.013	1.103	1.198	1.877
	0.95	1.274	1.293	1.373	1.478	1.581	1.682	2.800
Policy DS		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.308	0.313	**0.336**	0.370	0.408	0.452	0.333
	0.50	0.348	0.356	0.390	**0.439**	0.496	0.561	0.400
	0.60	0.402	0.412	0.456	**0.520**	0.595	0.677	0.500
	0.70	0.483	0.495	0.547	0.623	**0.709**	0.803	0.667
	0.80	0.629	0.643	0.700	0.782	0.871	0.968	0.999
	0.85	0.763	0.776	0.834	0.915	1.002	1.094	1.321
	0.90	0.993	1.006	1.062	1.140	1.221	1.307	1.877
	0.95	1.494	1.506	1.560	1.638	1.722	1.807	2.800

As the utilization increases, the disadvantage of the DS policy becomes more significant. The difference between FT_{DS} and FT_{SPT} increases. The reason is that as utilization increases the average number of unsequenced jobs at departure times increases as in Figure 5.3. This situation decreases the probability that the smallest of sequenced jobs is the smallest among all jobs in the system.

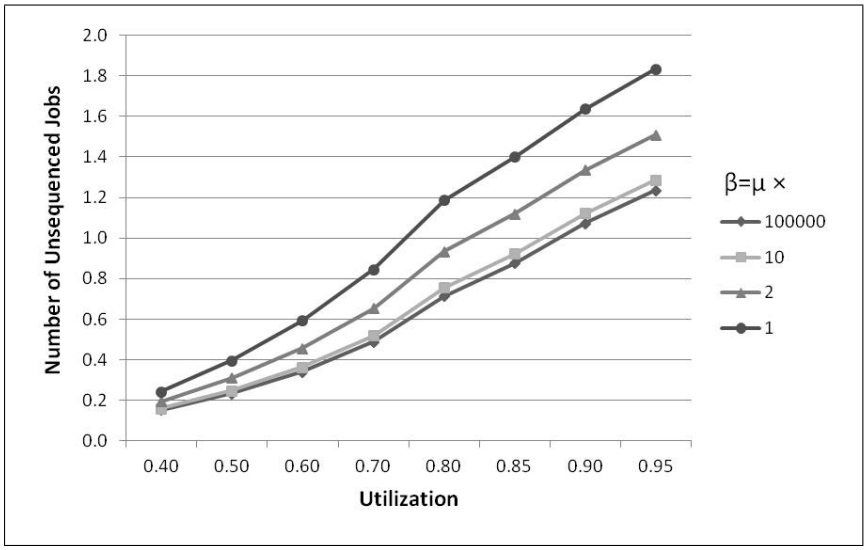
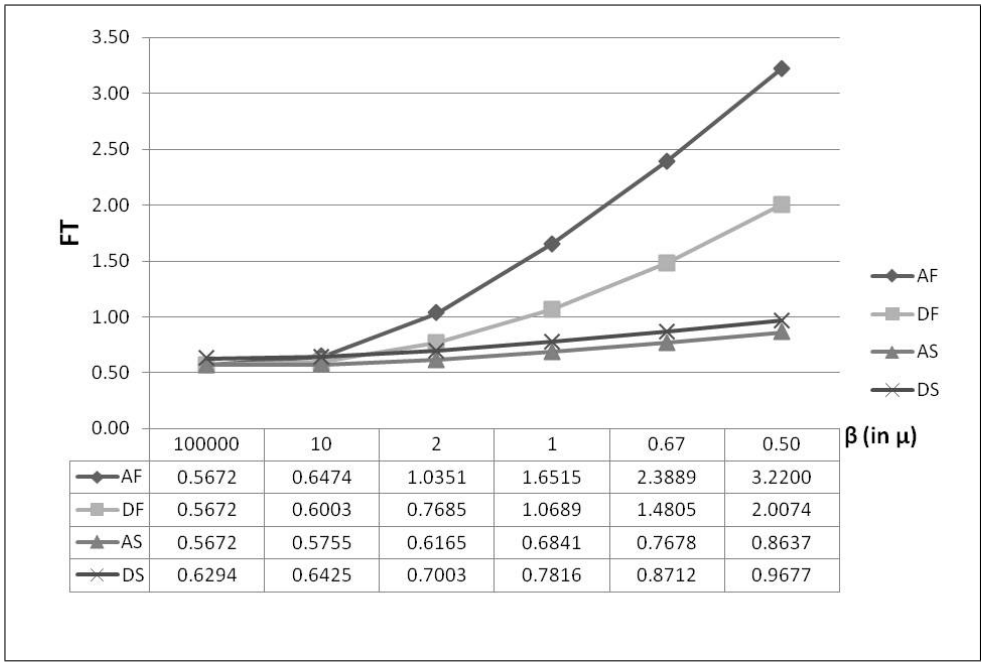


Figure 5.3. Average Number of Unsequenced Jobs Under the DS Policy.

It can be concluded that the AS policy performs better than the DS policy. For all β and ρ combinations, $FT_{AS} < FT_{DS}$. In addition, the number of experiments resulting in $FT > FT_{FCFS}$ is less in arrival triggered case. Remember that, the DF policy performs better than the AF policy in all experiments. Therefore it is obvious that the improvement due to continuation of services during scheduling is greater in arrival triggered case.

As expected, in both the AS and the DS policies, the increase in mean flow time as β decreases is very small compared to the F policies. Figure 5.4 presents the increase in mean flow time as β decreses under the F and the S policies, for $\rho = 0.80$. For the β set that is used, a significantly smaller number of experiments resulted in $FT > FT_{FCFS}$. Especially in higher utilization systems, even the smallest sequencing rate in the set resulted in $FT < FT_{FCFS}$. In systems with smaller utilizations, the break-even sequencing rate can still be observed. When they are compared to the results in Table 5.1, it is clear that the break-even β is smaller in the S policies. It should be underlined that there is still a break-even point. This is caused by keeping the server idle during long processing durations even there are some jobs in the system.



	100000	10	2	1	0.67	0.50
AF	0.5672	0.6474	1.0351	1.6515	2.3889	3.2200
DF	0.5672	0.6003	0.7685	1.0689	1.4805	2.0074
AS	0.5672	0.5755	0.6165	0.6841	0.7678	0.8637
DS	0.6294	0.6425	0.7003	0.7816	0.8712	0.9677

Figure 5.4. Mean Flow Time for $\rho = 0.80$.

In the I policies, the server is never kept idle unless the system is empty. The resulting mean flow times are presented in Table 5.3. As in the previous cases the first part to examine is $\beta = 100000$ columns which is expected to be equal to FT_{SPT}. $FT_{AI} = FT_{SPT}$ holds as expected. However the results for departure triggered case is remarkable. They are larger compared to the DS policy. The disadvantage of sequencing at departures has increased. The difference between DI and DS policies is that if there are not any sequenced jobs in the system, the server continues processing in random order under the DI policy. Since this is the only difference, this must be the reason behind this increase of mean flow time for departure triggered case.

To explain the result $FT_{DI} > FT_{DS}$ for $\beta = 100000$, consider the state $(2, 0, sp)$. This is a recurrent state having positive long run probability under both DS and DI policies. However it does not mean the same thing for both of them. In the DS policy, since the number of sequenced jobs is zero, the server waits for sequencing completion in this state. Since the sequencing time is zero, there is no cost of waiting but there is the benefit of sequencing in that case. In the DI policy, at state $(2, 0, sp)$ the server picks the first job in the queue and when it starts its service there is only one job left

in the queue. The benefit from sequencing is lost since the server does not prefer to wait for a very short sequencing duration. This is the reason behind the small increase in FT for $\beta = 100000$ compared to FT_{DS} which is also greater than FT_{SPT}.

Table 5.3. The I Policy Experimental Results.

Policy AI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.307	0.309	0.311	0.312	0.313	0.333
	0.50	0.344	0.345	0.350	0.355	0.359	0.362	0.400
	0.60	0.342	0.392	0.403	0.413	0.420	0.427	0.500
	0.70	0.454	0.459	0.476	0.494	0.508	0.519	0.667
	0.80	0.567	0.574	0.602	0.631	0.653	0.672	0.999
	0.85	0.669	0.678	0.714	0.752	0.782	0.807	1.321
	0.90	0.849	0.861	0.912	0.968	1.013	1.050	1.877
	0.95	1.274	1.292	1.367	1.461	1.547	1.624	2.800
Policy DI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.316	0.316	0.317	0.318	0.318	0.319	0.333
	0.50	0.367	0.368	0.371	0.373	0.375	0.376	0.400
	0.60	0.434	0.436	0.442	0.449	0.454	0.458	0.500
	0.70	0.525	0.529	0.541	0.554	0.565	0.575	0.667
	0.80	0.669	0.674	0.692	0.715	0.736	0.754	0.999
	0.85	0.793	0.798	0.820	0.848	0.876	0.901	1.321
	0.90	1.010	1.016	1.043	1.082	1.120	1.156	1.877
	0.95	1.509	1.518	1.561	1.625	1.694	1.763	2.800

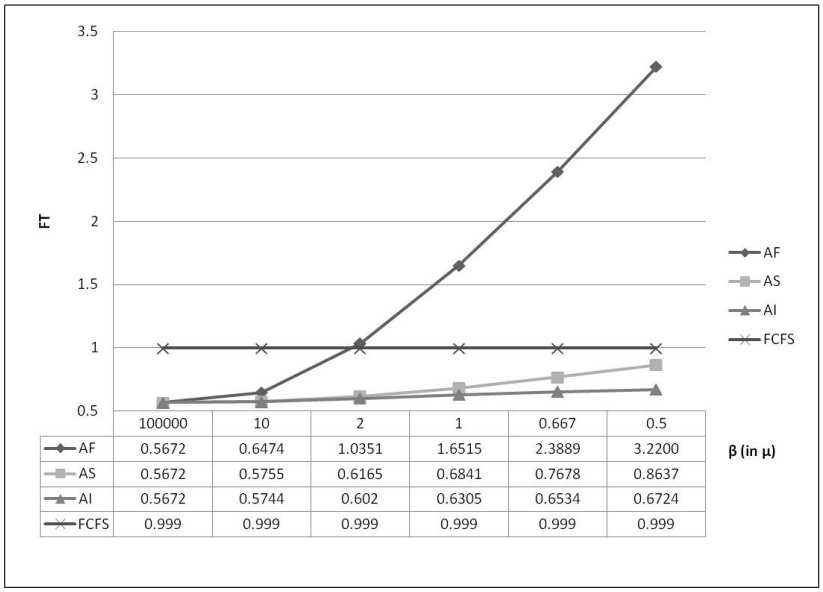
As β decreases mean flow time increases even slower than the S policies. The I rule is expected to improve the performance especially for small β. In other words, the idea behind considering this policy is that if the sequencing time is very long, random order processing may be a better choice than waiting. So the number in the last columns of the Table 5.3 are the actual indicators of the favorableness of this policy. It is seen that they are smaller than the previous policies. It can be concluded that for long expected sequencing times, sequencing is not worth to keep the server idle.

Utilization increase increases the benefit from sequencing in terms of mean flow time, as in the F and S policies. An opposing idea against this policy could be that, as utilization increases, the benefit from sequencing becomes so significant that waiting idle for such a sequence may be reasonable. However its is seen that for high utilizations as well, waiting idle is not advantageous. Let's say there are k unsequenced jobs in the system. Expectation of this number is larger in higher utilizations. In this last policy, if there are not any sequenced jobs, picking up one to serve and sequencing the remaining $k-1$ seems still very beneficial. Hence, this policy deals with long τ for all utilizations fairly well.

Both AI and DI policies give better mean flow times than FCFS discipline. There are not any bold numbers in Table 5.3 which indicates that within the boundaries of our test parameter sets the systems always perform better than FCFS. This is an expected result since the worst case scenario under I policies is random order processing. As long as there are some sequenced jobs, systems benefit from sequencing. Otherwise they perform as good as FCFS. Sequencing even two jobs causes some benefit. Hence sequencing in some states and random processing in others never results in worse than random processing all the time.

If we focus on the A policies only, we see that freezing the server always incurs more time cost than not freezing. For all β and ρ, largest mean flow times are obtained under the AF policy. The AS policy improves the performance significantly. Although the AI policy results better than the AS policy for small β, the difference between is not very large. As an example, AF, AS and AI policies are compared for $\rho = 0.80$ in Figure 5.5(a). AS and AI policies are close to each other. Note that for zero sequencing time all policies have the same result.

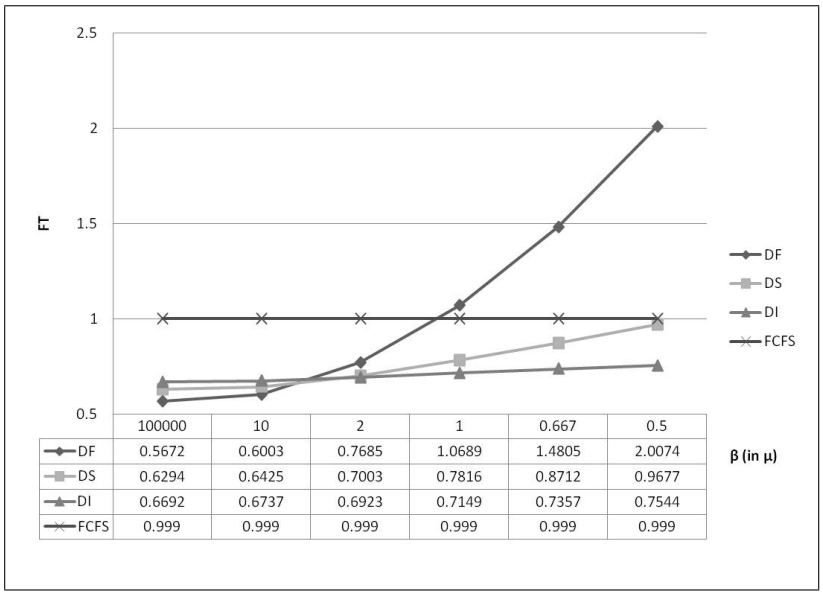
Under the D policies the situation is different. For $\tau = 0$, freeze the server rule performs better than the other rules. The reason is the disadvantage mentioned previously. When there are two jobs at a service completion moment, the server picks one and this prevents sequencing since there is one job left in the queue. As expected, it is not reasonable not to wait for sequencing when it is not time consuming.



	100000	10	2	1	0.667	0.5
AF	0.5672	0.6474	1.0351	1.6515	2.3889	3.2200
AS	0.5672	0.5755	0.6165	0.6841	0.7678	0.8637
AI	0.5672	0.5744	0.602	0.6305	0.6534	0.6724
FCFS	0.999	0.999	0.999	0.999	0.999	0.999

β (in μ)

(a) A Policies



	100000	10	2	1	0.667	0.5
DF	0.5672	0.6003	0.7685	1.0689	1.4805	2.0074
DS	0.6294	0.6425	0.7003	0.7816	0.8712	0.9677
DI	0.6692	0.6737	0.6923	0.7149	0.7357	0.7544
FCFS	0.999	0.999	0.999	0.999	0.999	0.999

β (in μ)

(b) D Policies

Figure 5.5. Comparison of Set 2 Rules for $\rho = 0.80$.

It is seen in Figure 5.5(b) that this reasoning is valid for positive but small τ as well. For small sequencing time, freezing the server still performs better. It can be explained such that if sequencing time is much less than processing time on the average, it is worth waiting idle. As β decreases, at some point the F rule loses its advantage. FT_{DI} is slightly greater than FT_{DS} for $\tau \in \{0, 0.1\mu\}$. Then, as τ increases the DI policy starts to perform better. Same situation is observed for all utilizations

under the D policies, comparing the results in Table 5.2 and Table 5.3.

The optimal sequencing policy depends on the physical capabilities of the system. If it is allowed to sequence during processing, then sequencing upon arrivals and continuing processing as long as the system is not empty results in the minimum mean flow times. Hence, it is the optimal policy. However if sequencing and processing cannot be carried out in parallel, one should choose to sequence at service completions if there is a positive time cost of sequencing.

If it is decided to sequence at departures for some external reason in a system which allows to carry out two operations in parallel, then the optimal decision is a function of ρ and β. For smaller β than the mean service rate, the DI policy is optimal; whereas for very large β the DF policy is the most beneficial policy.

6. CONCLUSION

In this thesis, we investigated the effect of sequencing time on the mean flow time in single server system under different sequencing policies. We addressed the question whether it is worth scheduling or not. For this purpose, as a test bed we considered a single server system where jobs arrive at the system following a Poisson distribution with rate λ and each job has a random processing time following an independent exponential distribution with rate μ. Jobs are sequenced by a scheduler in another independent exponentially distribution random time with rate β following SPT rule.

In the first part of our study, we modeled this system as a CTMC using state dependent service rates in order to imitate an M/M/1/SPT queue. Service rates in the MC were obtained empirically as a function of utilization, original service rate μ and the number of sequenced jobs is the queue. Using this service rate approximation, the system was modeled under two main sequencing trigger point decisions.

The first sequencing policy is sequencing the jobs upon arrivals. In this policy, every arrival starts a sequencing and the scheduler sequences the jobs in the queue according to SPT rule. There are three alternative policies under this policy. The first one is freezing the server during sequencing. The server waits for sequencing completion to continue its service. Alternatively, the server continues processing during sequencing. If sequencing does not finish until its service completion, it continues processing the previously sequenced jobs in the system. In the last policy, the server continues processing unless the system is empty.

The second trigger point considered is a service completion. The same alternatives as the arrival triggered policies, regarding the server behavior during sequencing, are considered here as well. The transition matrices are generated for all policies. Then solving the balance equations the mean queue length and the mean flow times are obtained for a set of sequencing time and system utilization combinations.

Results show that, the effect of sequencing time changes under each policy. Especially, in freeze the server during sequencing policies, decision time has a more significant effect than the other policies. If the system features does not allow to carry out sequencing and processing in parallel, one should prefer to sequence upon departures for which longer sequencing times can be tolerated. Otherwise sequencing efforts become worthless, since the expected flow time increases significantly with increasing sequencing time. If both operations can be conducted in parallel, sequencing upon arrivals is more beneficial for the aim of minimizing expected flow time in the system, and the sequencing should not be carried at the expense of holding the server idle.

Another observation is that the system utilization also affects the benefit from sequencing in terms of mean flow time in the existence of positive decision time. Since in our study we consider a system with finite queue capacity, at some point utilization increase does not increase the benefit from sequencing any more. Unfortunately, there are not many studies in the literature on systems with time cost of scheduling. Therefore, we can evaluate the results of our policies only comparing them to the other results that we obtain.

One of the future work ideas is to study the case that the scheduler does not realize system state changes during scheduling. It takes a snapshot of the queue, starts optimizing the sequence and when it completes sequencing the system may have changed. In such a situation, the resulting sequence becomes suboptimal. This policy can be compared to the policies studied in this thesis.

As a different dispatching policy, a critical number of new jobs in the queue can be introduced. When there are more unsequenced jobs than the critical number, the queue is sequenced, otherwise the jobs are served under FCFS rule. In our models, the sequencing time is assumed to be load independent. However, it may be assumed to be increasing with the number of jobs to sequence. Hence, in further researches, load dependent sequencing times can be considered as well.

APPENDIX A: SERVICE RATE APPROXIMATION

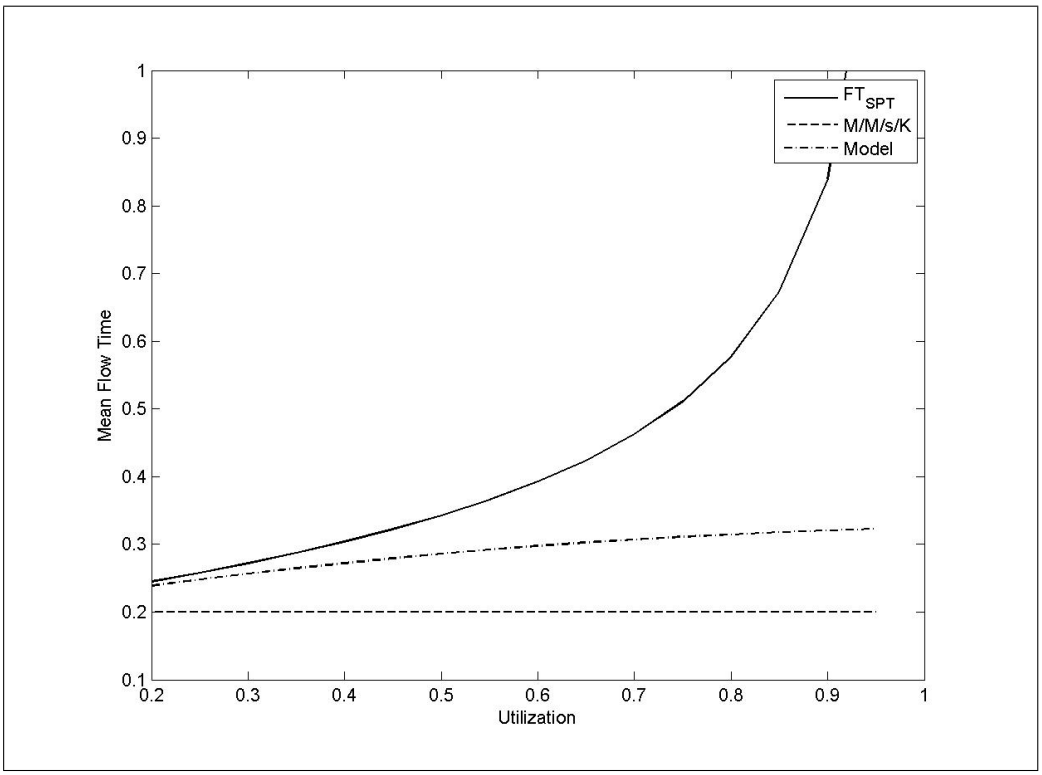


Figure A.1. SPT vs. M/M/s/K Approximations.

Note: Model refers to the service rate approximation $\mu_t = (X_1(t)+1)\mu$ and in M/M/s/K model $s = K = 100$.

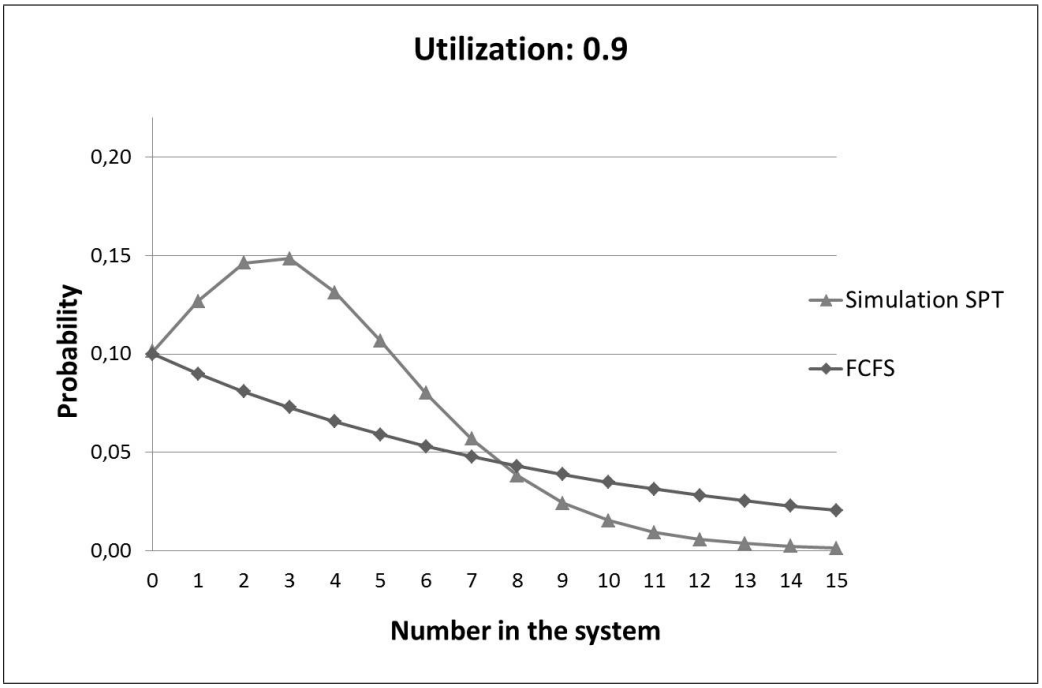


Figure A.2. Steady State Probabilities Simulation vs. Model Estimate.

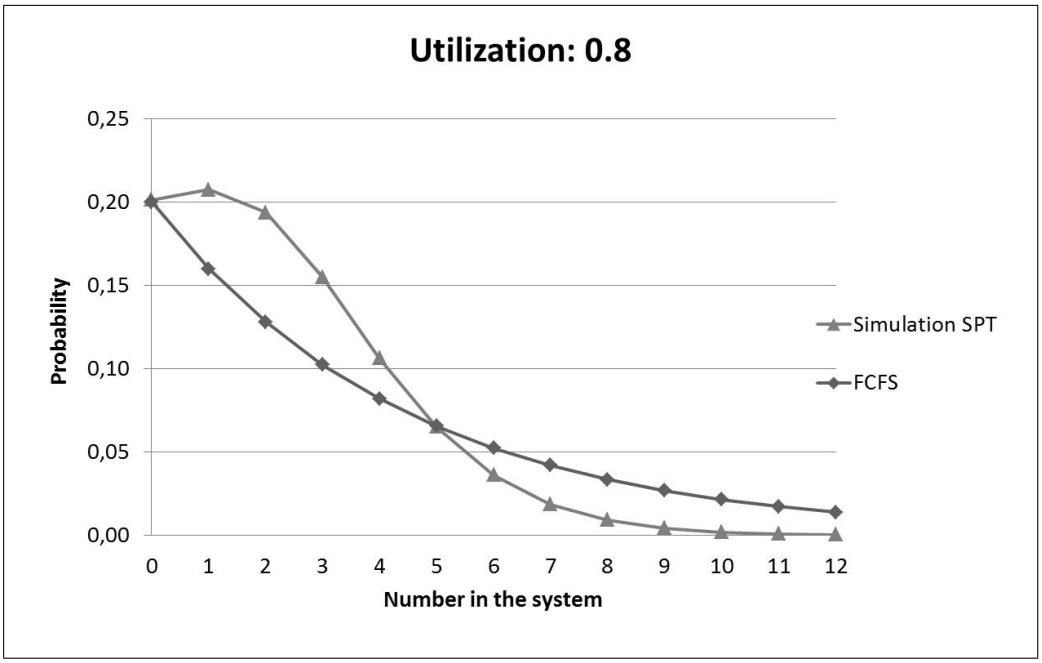


Figure A.3. Steady State Probabilities Simulation vs. Model Estimate.

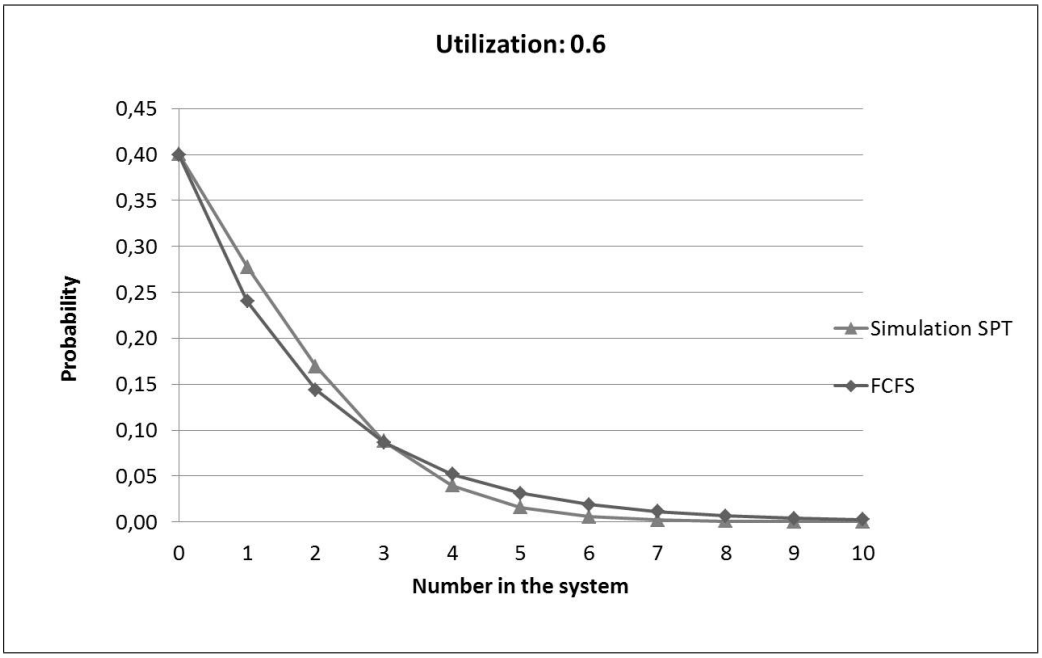


Figure A.4. Steady State Probabilities Simulation vs. Model Estimate.

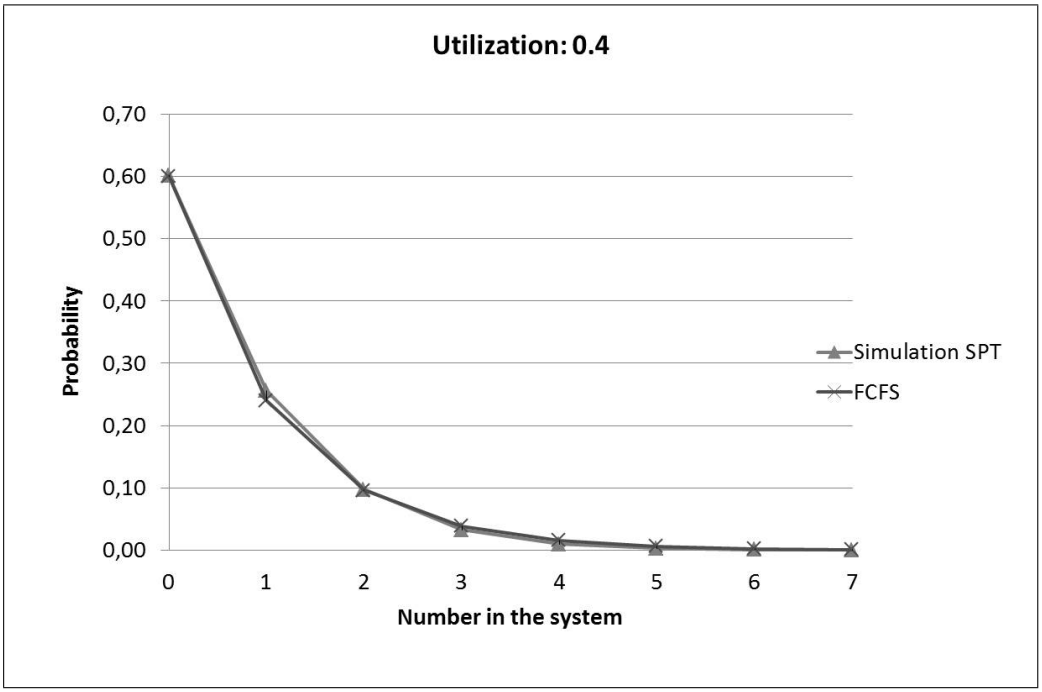


Figure A.5. Steady State Probabilities Simulation vs. Model Estimate.

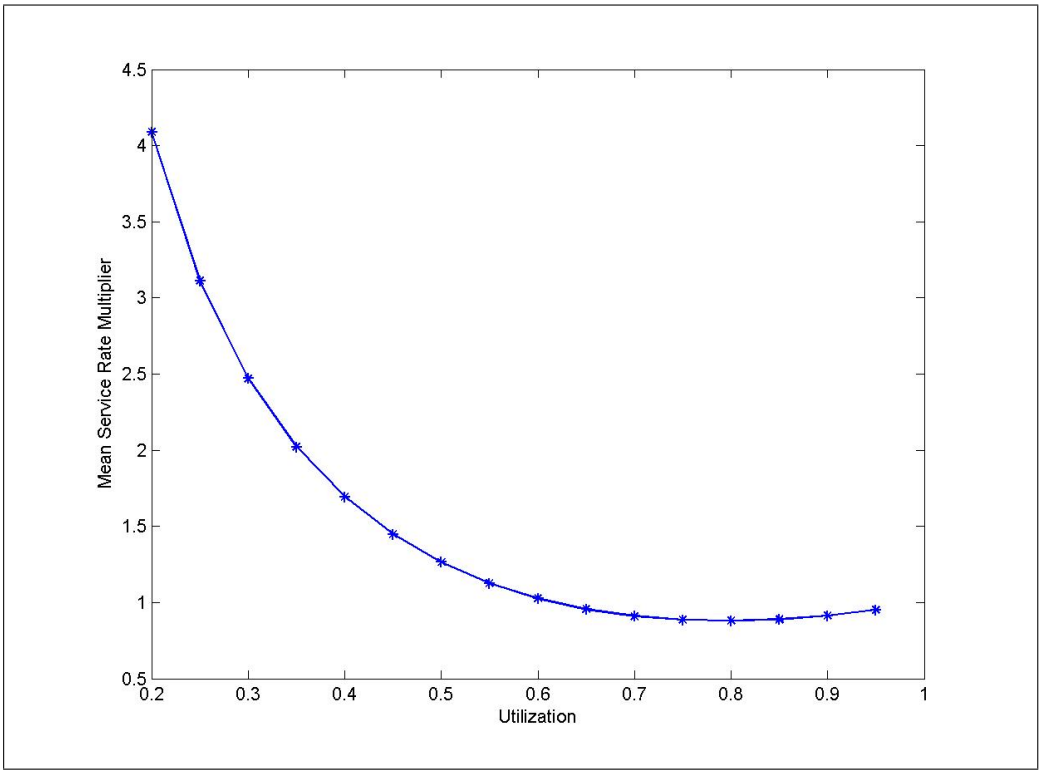


Figure A.6. Mean Service Rate Multiplier of Model 1.

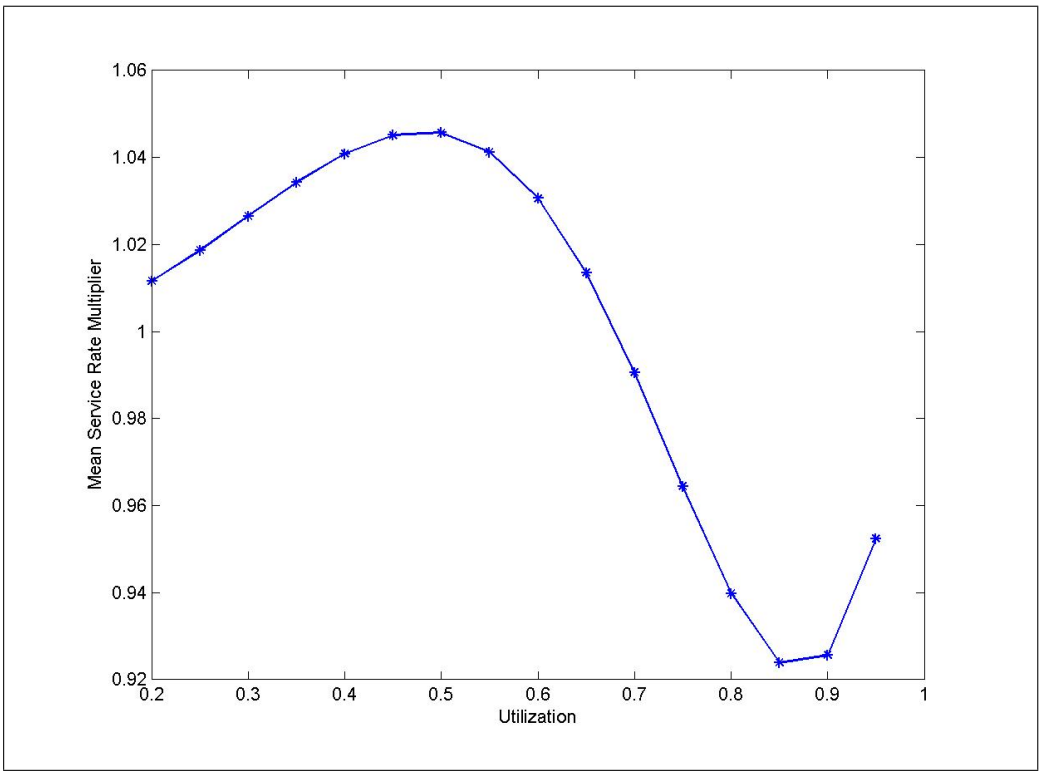


Figure A.7. Mean Service Rate Multiplier of Model 2.

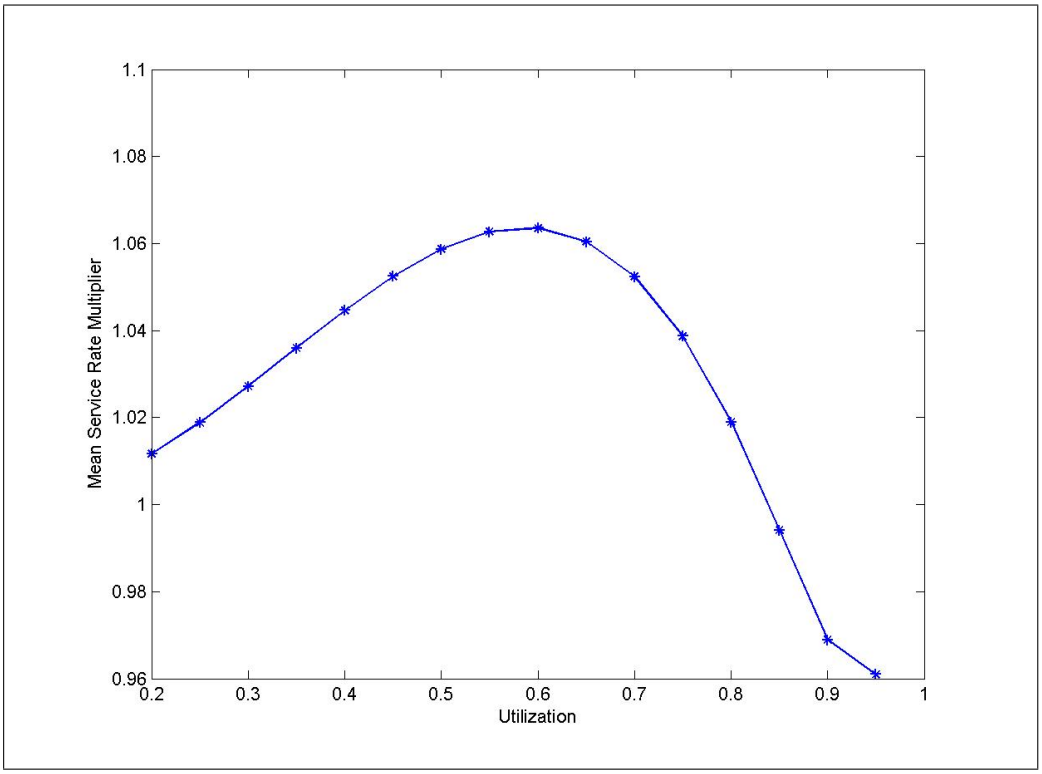


Figure A.8. Mean Service Rate Multiplier of Model 3.

APPENDIX B: EXPERIMENTAL RESULTS

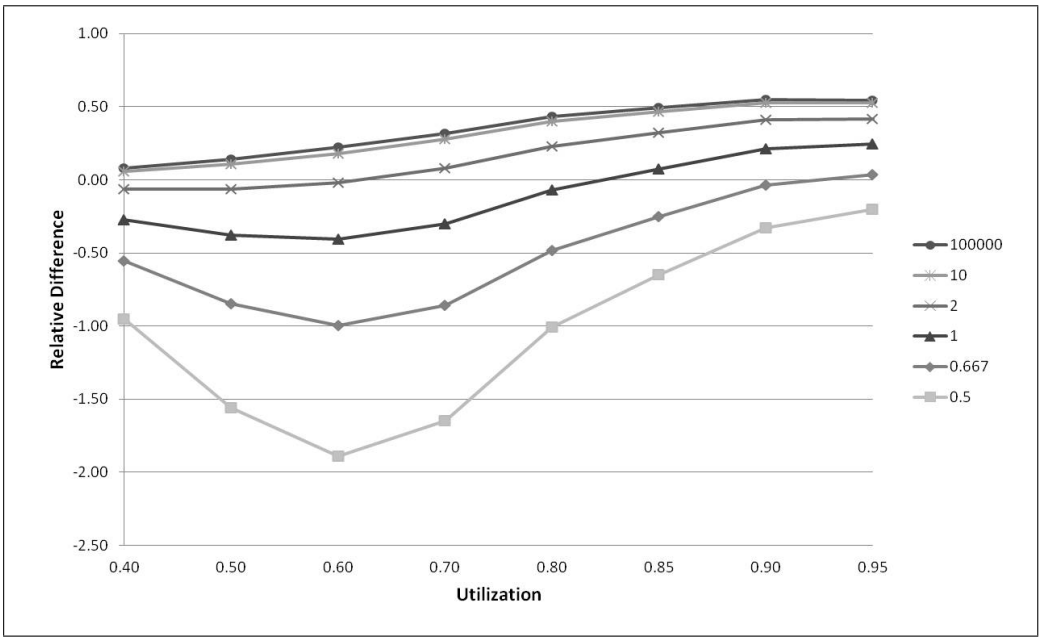


Figure B.1. Relative Difference Under the DF Policy.

Table B.1. The F Policy Experimental Results C=20.

Policy AF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.323	**0.403**	0.546	0.755	1.052	0.333
	0.50	0.344	0.372	**0.517**	0.800	1.236	1.833	0.400
	0.60	0.390	0.432	**0.656**	1.095	1.720	2.443	0.500
	0.70	0.454	0.513	**0.819**	1.365	2.029	2.678	0.663
	0.80	0.567	0.647	**1.034**	1.627	2.232	2.751	0.951
	0.85	0.669	0.764	**1.197**	1.800	2.363	2.817	1.165
	0.90	0.849	0.964	**1.456**	2.073	2.588	2.970	1.427
	0.95	1.274	1.428	**2.037**	2.665	3.081	3.349	1.717
Policy DF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.315	**0.355**	0.424	0.519	0.650	0.333
	0.50	0.344	0.357	**0.426**	0.551	0.739	1.013	0.400
	0.60	0.390	0.409	**0.510**	0.702	0.994	1.403	0.500
	0.70	0.454	0.480	0.614	**0.865**	1.228	1.689	0.663
	0.80	0.567	0.600	0.769	**1.068**	1.463	1.909	0.951
	0.85	0.669	0.707	0.895	**1.220**	1.626	2.055	1.165
	0.90	0.849	0.893	1.111	**1.473**	1.895	2.303	1.427
	0.95	1.274	1.333	1.626	**2.076**	2.512	2.855	1.717

Table B.2. The F Policy Experimental Results C=30.

Policy AF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.323	**0.403**	0.546	0.757	1.072	0.333
	0.50	0.344	0.372	**0.517**	0.802	1.265	1.988	0.400
	0.60	0.390	0.432	**0.656**	1.104	1.805	2.782	0.500
	0.70	0.454	0.513	**0.819**	1.380	2.150	3.067	0.667
	0.80	0.567	0.647	**1.035**	1.651	2.374	3.132	0.992
	0.85	0.669	0.764	1.198	**1.834**	2.529	3.219	1.286
	0.90	0.849	0.964	1.461	**2.133**	2.822	3.463	1.727
	0.95	1.274	1.429	2.062	**2.867**	3.604	4.187	2.328
Policy DF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.315	**0.355**	0.424	0.519	0.650	0.333
	0.50	0.344	0.357	**0.426**	0.551	0.740	1.023	0.400
	0.60	0.390	0.409	**0.510**	0.702	0.999	1.442	0.500
	0.70	0.454	0.480	0.614	**0.866**	1.239	1.757	0.667
	0.80	0.567	0.600	0.769	**1.069**	1.480	1.999	0.992
	0.85	0.669	0.707	0.895	1.222	**1.650**	2.166	1.286
	0.90	0.849	0.893	1.111	1.478	**1.942**	2.472	1.727
	0.95	1.274	1.334	1.628	2.111	**2.687**	3.279	2.328

Table B.3. The F Policy Experimental Results C=50.

Policy AF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.323	**0.403**	0.546	0.757	1.074	0.333
	0.50	0.344	0.372	**0.517**	0.802	1.268	2.025	0.400
	0.60	0.390	0.432	**0.656**	1.104	1.815	2.889	0.500
	0.70	0.454	0.513	**0.819**	1.381	2.165	3.186	0.667
	0.80	0.567	0.647	**1.035**	1.652	2.390	3.233	1.000
	0.85	0.669	0.764	1.198	**1.835**	2.549	3.326	1.330
	0.90	0.849	0.964	1.461	**2.136**	2.855	3.611	1.947
	0.95	1.274	1.429	2.063	2.884	**3.733**	4.595	3.153
Policy DF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.315	**0.355**	0.424	0.519	0.650	0.333
	0.50	0.344	0.357	**0.426**	0.551	0.740	1.024	0.400
	0.60	0.390	0.409	**0.510**	0.702	0.999	1.446	0.500
	0.70	0.454	0.480	0.614	**0.866**	1.239	1.764	0.667
	0.80	0.567	0.600	0.769	**1.069**	1.481	2.008	1.000
	0.85	0.669	0.707	0.895	1.222	**1.651**	2.178	1.330
	0.90	0.849	0.893	1.111	1.478	1.944	**2.493**	1.947
	0.95	1.274	1.334	1.628	2.111	2.702	**3.375**	3.153

Table B.4. The F Policy Experimental Results C=60.

Policy AF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.323	**0.403**	0.546	0.757	1.074	0.333
	0.50	0.344	0.372	**0.517**	0.802	1.268	2.025	0.400
	0.60	0.390	0.432	**0.656**	1.104	1.815	2.892	0.500
	0.70	0.454	0.513	**0.819**	1.381	2.165	3.188	0.667
	0.80	0.567	0.647	**1.035**	1.652	2.390	3.235	1.000
	0.85	0.669	0.764	1.198	**1.835**	2.549	3.327	1.333
	0.90	0.849	0.964	1.461	**2.136**	2.855	3.613	1.978
	0.95	1.274	1.429	2.063	2.884	**3.734**	4.607	3.412
Policy DF		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.315	**0.355**	0.424	0.519	0.650	0.333
	0.50	0.344	0.357	**0.426**	0.551	0.740	1.024	0.400
	0.60	0.390	0.409	**0.510**	0.702	0.999	1.446	0.500
	0.70	0.454	0.480	0.614	**0.866**	1.239	1.764	0.667
	0.80	0.567	0.600	0.769	**1.069**	1.481	2.008	1.000
	0.85	0.669	0.707	0.895	1.222	**1.651**	2.178	1.333
	0.90	0.849	0.893	1.111	1.478	1.944	**2.493**	1.978
	0.95	1.274	1.334	1.628	2.111	2.702	3.376	3.412

Table B.5. The S Policy Experimental Results C=20.

Policy AS		β						FT_{FCFS}
		100,000	10	2	1	0.667	0.5	
ρ	0.40	0.306	0.307	0.316	**0.337**	0.366	0.403	0.333
	0.50	0.344	0.345	0.358	0.390	**0.435**	0.491	0.400
	0.60	0.390	0.392	0.412	0.455	**0.516**	0.590	0.500
	0.70	0.454	0.459	0.487	0.542	0.617	**0.705**	0.663
	0.80	0.567	0.576	0.617	0.684	0.768	0.864	0.951
	0.85	0.669	0.680	0.731	0.806	0.892	0.989	1.165
	0.90	0.849	0.864	0.928	1.013	1.103	1.197	1.427
	0.95	1.273	1.292	1.373	1.476	1.578	1.677	1.717
Policy DS		β						FT_{FCFS}
		100,000	10	2	1	0.667	0.5	
ρ	0.40	0.308	0.313	**0.336**	0.370	0.408	0.452	0.333
	0.50	0.348	0.356	0.390	**0.439**	0.496	0.561	0.400
	0.60	0.402	0.412	0.456	**0.520**	0.595	0.677	0.500
	0.70	0.483	0.495	0.547	0.623	**0.709**	0.803	0.663
	0.80	0.629	0.643	0.700	0.782	0.871	**0.968**	0.951
	0.85	0.763	0.776	0.834	0.914	1.001	1.094	1.165
	0.90	0.993	1.006	1.062	1.139	1.221	1.306	1.427
	0.95	1.492	1.503	1.557	1.634	1.693	1.702	1.717

Table B.6. The S Policy Experimental Results C=30.

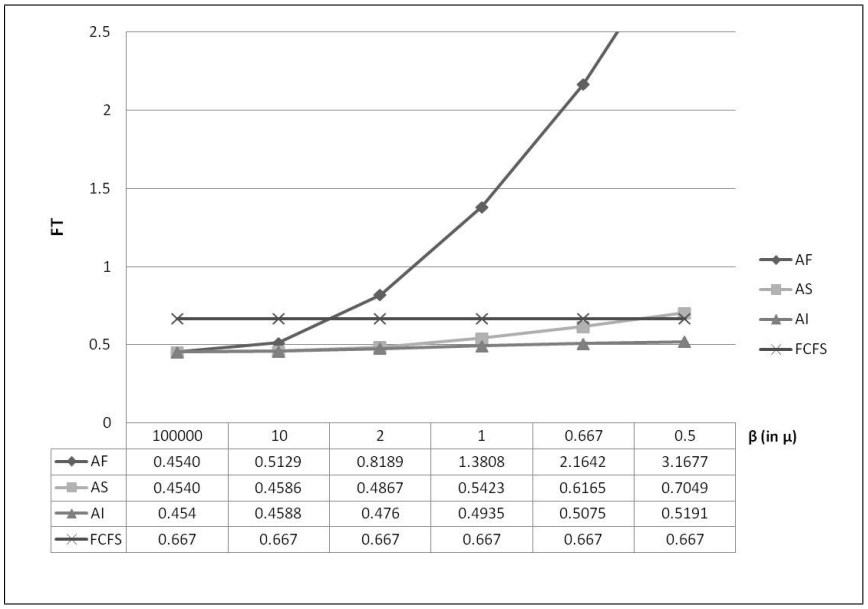
Policy AS		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.307	0.316	**0.337**	0.366	0.403	0.333
	0.50	0.344	0.345	0.358	0.390	**0.435**	0.491	0.400
	0.60	0.390	0.392	0.412	0.455	**0.516**	0.590	0.500
	0.70	0.454	0.459	0.487	0.542	0.617	**0.705**	0.667
	0.80	0.567	0.576	0.617	0.684	0.768	0.864	0.992
	0.85	0.437	0.680	0.731	0.806	0.893	0.989	1.286
	0.90	0.849	0.864	0.928	1.013	1.103	1.198	1.727
	0.95	1.274	1.293	1.373	1.478	1.581	1.682	2.328
Policy DS		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.308	0.313	**0.336**	0.370	0.408	0.452	0.333
	0.50	0.348	0.356	0.390	**0.439**	0.496	0.561	0.400
	0.60	0.402	0.412	0.456	**0.520**	0.595	0.677	0.500
	0.70	0.483	0.495	0.547	0.623	**0.709**	0.803	0.667
	0.80	0.629	0.643	0.700	0.782	0.871	0.968	0.992
	0.85	0.763	0.776	0.834	0.915	1.002	1.094	1.286
	0.90	0.993	1.006	1.062	1.140	1.221	1.307	1.727
	0.95	1.494	1.506	1.560	1.638	1.721	1.806	2.328

Table B.7. The I Policy Experimental Results C=20.

Policy AI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.307	0.309	0.311	0.312	0.313	0.333
	0.50	0.344	0.345	0.350	0.355	0.359	0.362	0.400
	0.60	0.390	0.392	0.403	0.413	0.420	0.427	0.500
	0.70	0.454	0.459	0.476	0.494	0.508	0.519	0.663
	0.80	0.567	0.574	0.602	0.631	0.653	0.672	0.951
	0.85	0.669	0.678	0.714	0.752	0.782	0.807	1.165
	0.90	0.849	0.861	0.912	0.968	1.013	1.050	1.427
	0.95	1.273	1.291	1.367	1.460	1.545	1.619	1.717
Policy DI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.316	0.316	0.317	0.318	0.318	0.319	0.333
	0.50	0.367	0.368	0.371	0.373	0.375	0.376	0.400
	0.60	0.434	0.436	0.442	0.449	0.454	0.458	0.500
	0.70	0.525	0.529	0.541	0.554	0.565	0.575	0.663
	0.80	0.669	0.674	0.692	0.715	0.736	0.754	0.951
	0.85	0.793	0.797	0.820	0.848	0.875	0.901	1.165
	0.90	1.010	1.015	1.043	1.081	1.119	1.154	1.427
	0.95	1.503	1.512	1.555	1.618	1.684	1.709	1.717

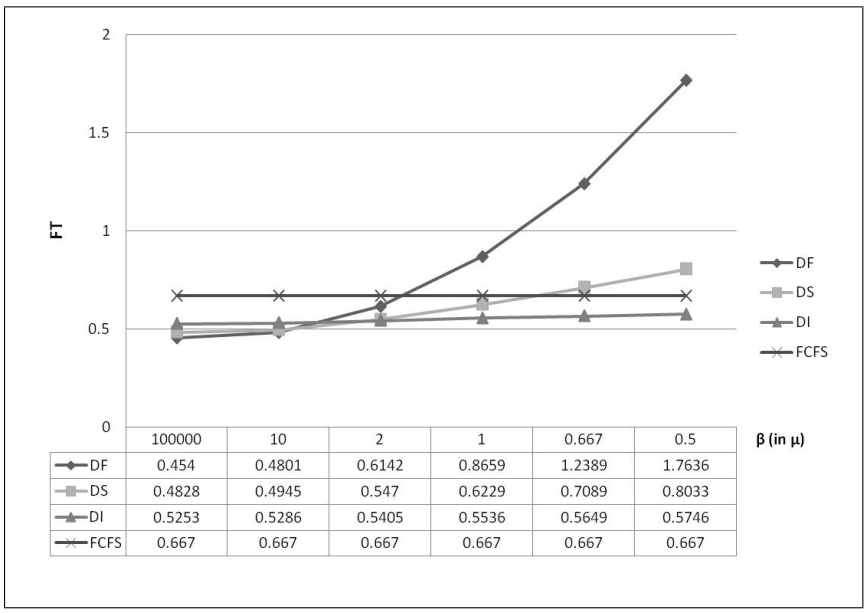
Table B.8. The I Policy Experimental Results C=30.

Policy AI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.306	0.307	0.309	0.311	0.312	0.313	0.333
	0.50	0.344	0.345	0.350	0.355	0.359	0.362	0.400
	0.60	0.390	0.392	0.403	0.413	0.420	0.427	0.500
	0.70	0.454	0.459	0.476	0.494	0.508	0.519	0.667
	0.80	0.567	0.574	0.602	0.631	0.653	0.672	0.992
	0.85	0.669	0.678	0.714	0.752	0.782	0.807	1.286
	0.90	0.849	0.861	0.912	0.968	1.013	1.050	1.727
	0.95	1.274	1.292	1.367	1.461	1.547	1.624	2.328
Policy DI		β						
		100,000	10	2	1	0.667	0.5	FT_{FCFS}
ρ	0.40	0.316	0.316	0.317	0.318	0.318	0.319	0.333
	0.50	0.367	0.368	0.371	0.373	0.375	0.376	0.400
	0.60	0.434	0.436	0.442	0.449	0.454	0.458	0.500
	0.70	0.525	0.529	0.541	0.554	0.565	0.575	0.667
	0.80	0.669	0.674	0.692	0.715	0.736	0.754	0.992
	0.85	0.793	0.798	0.820	0.848	0.876	0.901	1.286
	0.90	1.010	1.016	1.043	1.082	1.120	1.156	1.727
	0.95	1.509	1.517	1.561	1.625	1.694	1.763	2.328



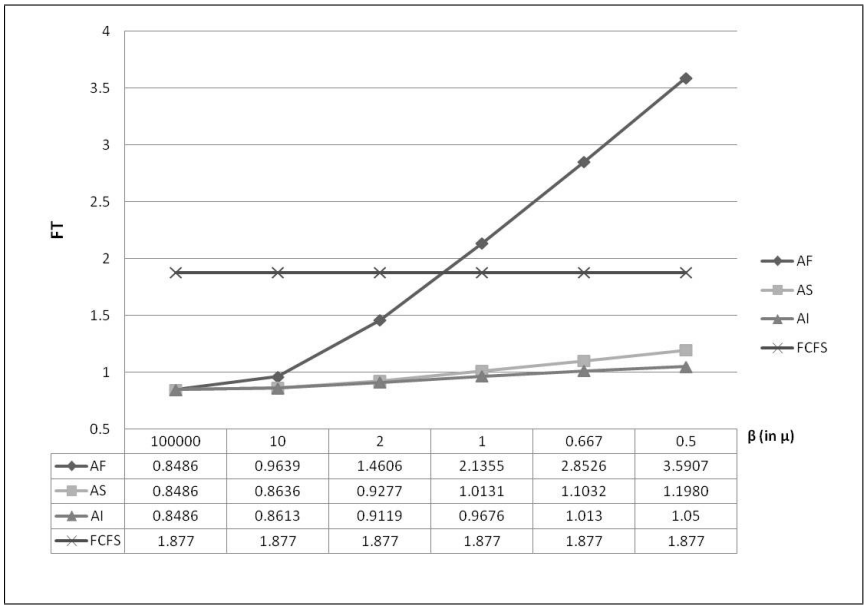
	100000	10	2	1	0.667	0.5	β (in μ)
AF	0.4540	0.5129	0.8189	1.3808	2.1642	3.1677	
AS	0.4540	0.4586	0.4867	0.5423	0.6165	0.7049	
AI	0.454	0.4588	0.476	0.4935	0.5075	0.5191	
FCFS	0.667	0.667	0.667	0.667	0.667	0.667	

Figure B.2. Mean Flow Time Under A Policies for $\rho = 0.70$.



	100000	10	2	1	0.667	0.5	β (in μ)
DF	0.454	0.4801	0.6142	0.8659	1.2389	1.7636	
DS	0.4828	0.4945	0.547	0.6229	0.7089	0.8033	
DI	0.5253	0.5286	0.5405	0.5536	0.5649	0.5746	
FCFS	0.667	0.667	0.667	0.667	0.667	0.667	

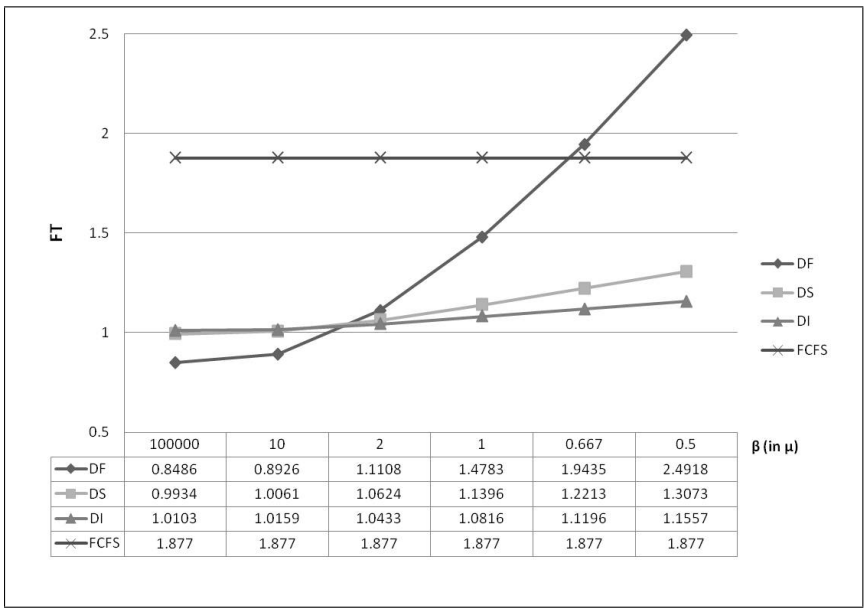
Figure B.3. Mean Flow Time Under D Policies for $\rho = 0.70$.



	100000	10	2	1	0.667	0.5
AF	0.8486	0.9639	1.4606	2.1355	2.8526	3.5907
AS	0.8486	0.8636	0.9277	1.0131	1.1032	1.1980
AI	0.8486	0.8613	0.9119	0.9676	1.013	1.05
FCFS	1.877	1.877	1.877	1.877	1.877	1.877

β (in μ)

Figure B.4. Mean Flow Time Under A Policies for $\rho = 0.90$.



	100000	10	2	1	0.667	0.5
DF	0.8486	0.8926	1.1108	1.4783	1.9435	2.4918
DS	0.9934	1.0061	1.0624	1.1396	1.2213	1.3073
DI	1.0103	1.0159	1.0433	1.0816	1.1196	1.1557
FCFS	1.877	1.877	1.877	1.877	1.877	1.877

β (in μ)

Figure B.5. Mean Flow Time Under D Policies for $\rho = 0.90$.

REFERENCES

1. Panwalkar, S. S. and W. Iskander, "A Survey of Scheduling Rules", *Operations Research*, Vol. 25, No. 1, pp. 45–61, 1977.

2. Jeong, K.-C. and Y.-D. Kim, "A Real-Time Scheduling Mechanism for a Flexible Manufacturing System: Using Simulation and Dispatching Rules", *International Journal of Production Research*, Vol. 36, No. 9, pp. 2609–2626, 1998.

3. Blackstone, J. H., D. T. Phillips and G. L. Hogg, "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations", *The International Journal of Production Research*, Vol. 20, No. 1, pp. 27–45, 1982.

4. Church, L. K. and R. Uzsoy, "Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops", *International Journal of Computer Integrated Manufacturing*, Vol. 5, No. 3, pp. 153–163, 1992.

5. Harris, C. M., "Queues with State-Dependent Stochastic Service Rates", *Operations Research*, Vol. 15, No. 1, pp. 117–130, 1967.

6. Sgall, J., "On-line scheduling", A. Fiat and G. Woeginger (Editors), *Online Algorithms*, Vol. 1442 of *Lecture Notes in Computer Science*, pp. 196–231, Springer Berlin Heidelberg, 1998.

7. Yih, Y. and A. Thesen, "Semi-Markov Decision Models for Real-Time Scheduling", *The International Journal of Production Research*, Vol. 29, No. 11, pp. 2331–2346, 1991.

8. Cobham, A., "Priority Assignment in Waiting Line Problems", *Journal of the Operations Research Society of America*, Vol. 2, No. 1, pp. 70–76, 1954.

9. Holley, J. L., "Letter to the Editor-Waiting Line Subject to Priorities", *Journal of*

the Operations Research Society of America, Vol. 2, No. 3, pp. 341–343, 1954.

10. Chang, W., "Queuing with Nonpreemptive and Preemptive-Resume Priorities", *Operations Research*, Vol. 13, No. 6, pp. 1020–1022, 1965.

11. Chang, W., "Preemptive Priority Queues", *Operations research*, Vol. 13, No. 5, pp. 820–827, 1965.

12. Phipps Jr, T. E., "Machine Repair as a Priority Waiting-Line Problem", *Operations Research*, Vol. 4, No. 1, pp. 76–85, 1956.

13. Conway, R. W. and W. L. Maxwell, "Network Dispatching by the Shortest-Operation Discipline", *Operations Research*, Vol. 10, No. 1, pp. 51–73, 1962.

14. Hawkes, A., "Time-Dependent Solution of a Priority Queue with Bulk Arrival", *Operations Research*, Vol. 13, No. 4, pp. 586–595, 1965.

15. Marks, B. I., "State Probabilities of M/M/1 Priority Queues", *Operations Research*, Vol. 21, No. 4, pp. 974–987, 1973.

16. Williams, T. M., "Nonpreemptive Multi-Server Priority Queues", *The Journal of the Operational Research Society*, Vol. 31, No. 12, pp. 1105–1107, 1980.

17. Kella, O. and U. Yechiali, "Waiting Times in the Non-Preemptive Priority M/M/c queue", *Stochastic Models*, Vol. 1, No. 2, pp. 257–262, 1985.

18. Schrage, L. E. and L. W. Miller, "The Queue M/G/1 with the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 14, No. 4, pp. 670–684, 1966.

19. Shanthikumar, J., "On Reducing Time Spent in M/G/1 Systems", *European Journal of Operational Research*, Vol. 9, No. 3, pp. 286–294, 1982.

20. Buzacott, J. A. and J. G. Shanthikumar, *Stochastic Models of Manufacturing Sys-*

tems, Prentice Hall, New Jersey, 1993.

21. Shanthikumar, J., "Comparison of Dispatch Policies for a Single Server Queueing Model with Limited Operational Control", *The International Journal of Production Research*, Vol. 22, No. 3, pp. 389–403, 1984.

22. Davis, W. J. and A. T. Jones, "A Real-Time Production Scheduler for a Stochastic Manufacturing Environment", *International Journal of Computer Integrated Manufacturing*, Vol. 1, No. 2, pp. 101–112, 1988.

23. Kim, M. H. and Y.-D. Kim, "Simulation-Based Real-Time Scheduling in a Flexible Manufacturing System", *Journal of manufacturing Systems*, Vol. 13, No. 2, pp. 85–93, 1994.

24. Avrahami, N. and Y. Azar, "Minimizing Total Flow Time and Total Completion Time with Immediate Dispatching", *Algorithmica*, Vol. 47, No. 3, pp. 253–268, 2007.

25. Sozer, K., *Analysis of the Impact of Decision Time on the System Performance in Distributed Systems*, Ph.D. Thesis, Bogazici University, 2005.

26. Oner, M., *The Value of Time Spent for Sequencing of Jobs in Random Sized Batches*, Master's Thesis, Bogazici University, 2012.

27. Kendall, D. G., "Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of the Imbedded Markov Chain", *The Annals of Mathematical Statistics*, Vol. 24, No. 3, pp. 338–354, 1953.

28. Miller Jr, R. G., "Priority Queues", *The Annals of Mathematical Statistics*, Vol. 31, No. 1, pp. 86–103, 1960.

29. Neuts, M. F., "Two Markov Chains, Arising from Examples of Queues with State-Dependent Service Times", *Sankhyā: The Indian Journal of Statistics, Series A*,

Vol. 29, No. 3, pp. 259–264, 1967.

30. Herzog, U., L. Woo and K. M. Chandy, "Solution of Queuing Problems by a Recursive Technique", *IBM Journal of Research and Development*, Vol. 19, No. 3, pp. 295–300, 1975.

31. Miller, D. R., "Computation of Steady-State Probabilities for M/M/1 Priority Queues", *Operations Research*, Vol. 29, No. 5, pp. 945–958, 1981.

32. Neuts, M. F., "Markov Chains with Applications in Queueing Theory, Which Have a Matrix-Geometric Invariant Probability Vector", *Advances in Applied Probability*, Vol. 10, No. 1, pp. 185–212, 1978.

33. Miller, D. R., *Steady-State Algorithmic Analysis of M/M/c Two-Priority Queues with Heterogeneous Rates*, Springer, 1982.

34. Kao, E. P. and K. S. Narayanan, "Computing Steady-State Probabilities of a Nonpreemptive Priority Multiserver Queue", *ORSA Journal on Computing*, Vol. 2, No. 3, pp. 211–218, 1990.

35. Conolly, B. and N. Hadidi, "A Correlated Queue", *Journal of Applied Probability*, Vol. 6, No. 1, pp. 122–136, 1969.

36. Hadidi, N. and B. Conolly, "On the Improvement of the Operational Characteristics of Single-Server Queues by the Use of a Queue-Length-Dependent Service Mechanism", *Applied Statistics*, Vol. 18, No. 3, pp. 229–240, 1969.

37. Courtois, P. and J. Georges, "On a Single-Server Finite Queuing Model with State-Dependent Arrival and Service Processes", *Operations Research*, Vol. 19, No. 2, pp. 424–435, 1971.

38. Cidon, I., R. Guréin, A. Khamisy and M. Sidi, "On Queues with Interarrival Times Proportional to Service Times", *Probability in the Engineering and Informational*

Sciences, Vol. 10, No. 01, pp. 87–107, 1996.

39. Robinson, L. W. and K. B. Hendricks, "Using State-Dependent Processing Rates to Emulate SPT Queue Discipline in an FCFS Queueing Network", *IIE transactions*, Vol. 27, No. 4, pp. 530–541, 1995.