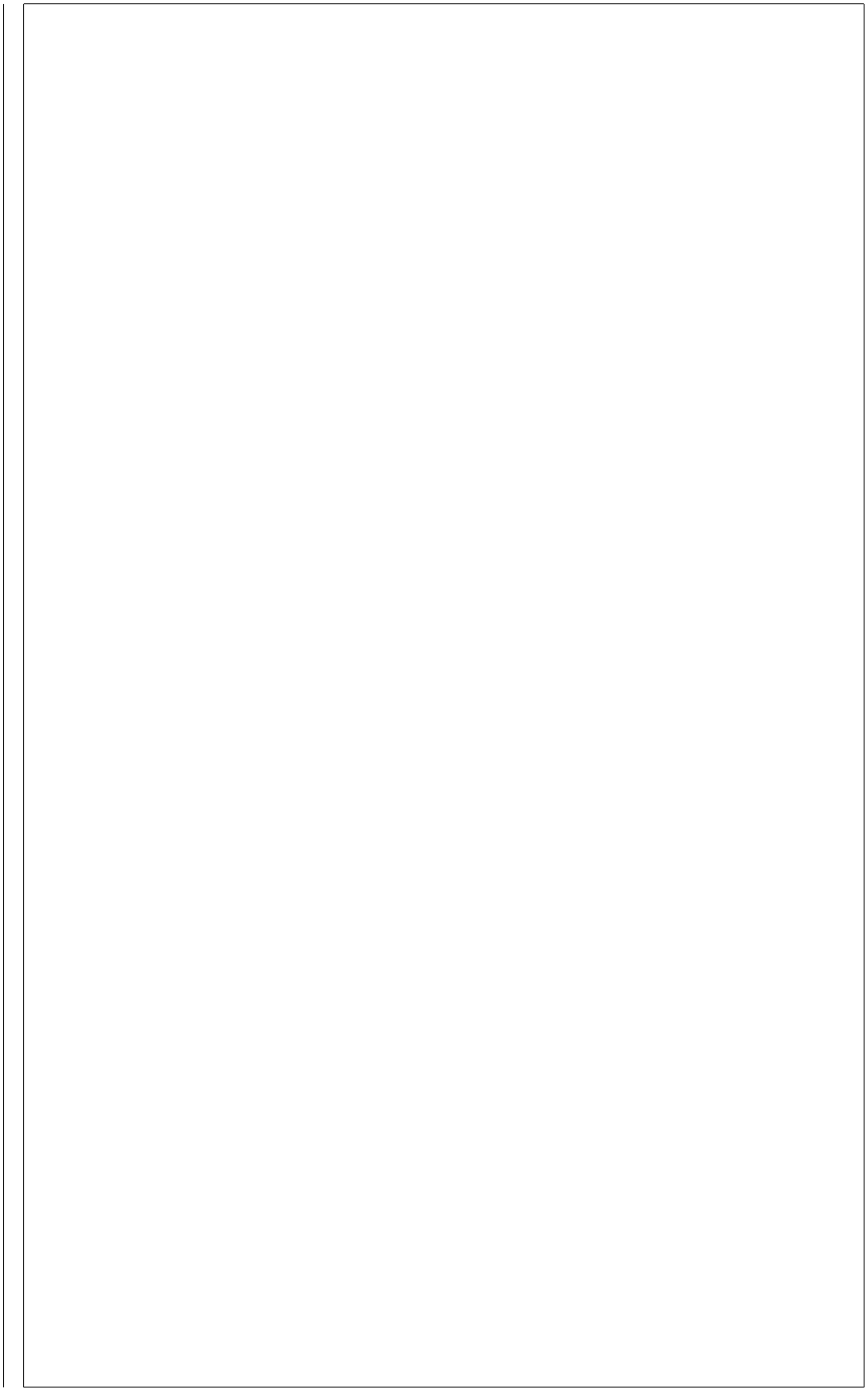
Queues with a Dynamic Schedule

John Gilbertson

A thesis presented for the degree of
Master of Science (Mathematics and Statistics)

Supervised by Professor Peter Taylor
Department of Mathematics and Statistics
The University of Melbourne
October 2016

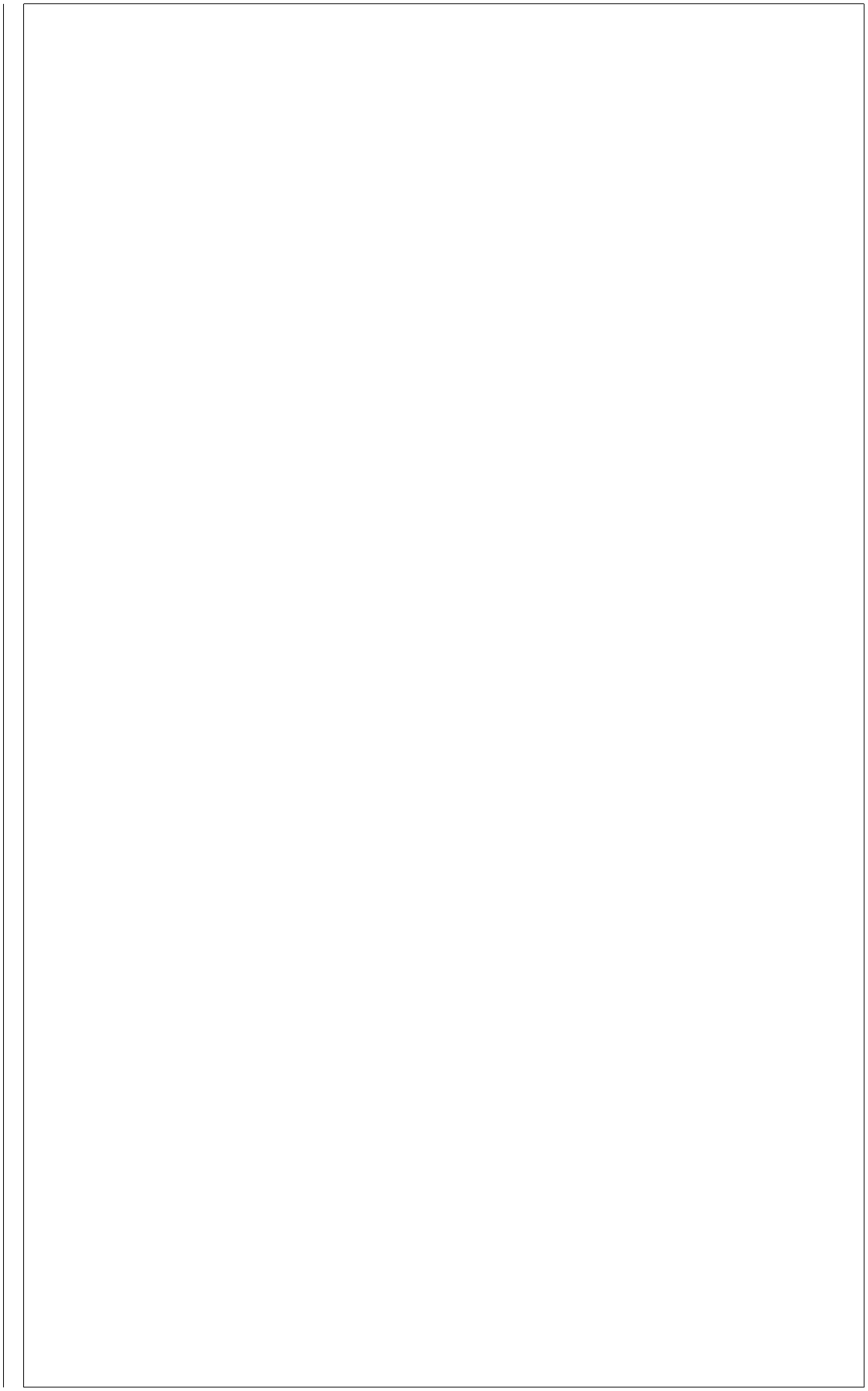


Declaration

This thesis is the sole work of the author whose name appears on the title page and it contains no material which the author has previously submitted for assessment at the University of Melbourne or elsewhere. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person, in the form of unacknowledged quotations or mathematical workings or in any other form, except where due reference is made. I declare that I have read, and in undertaking this research I have complied with, the University's Code of Conduct for Research. I also declare that I understand what is meant by plagiarism and that this is unacceptable.

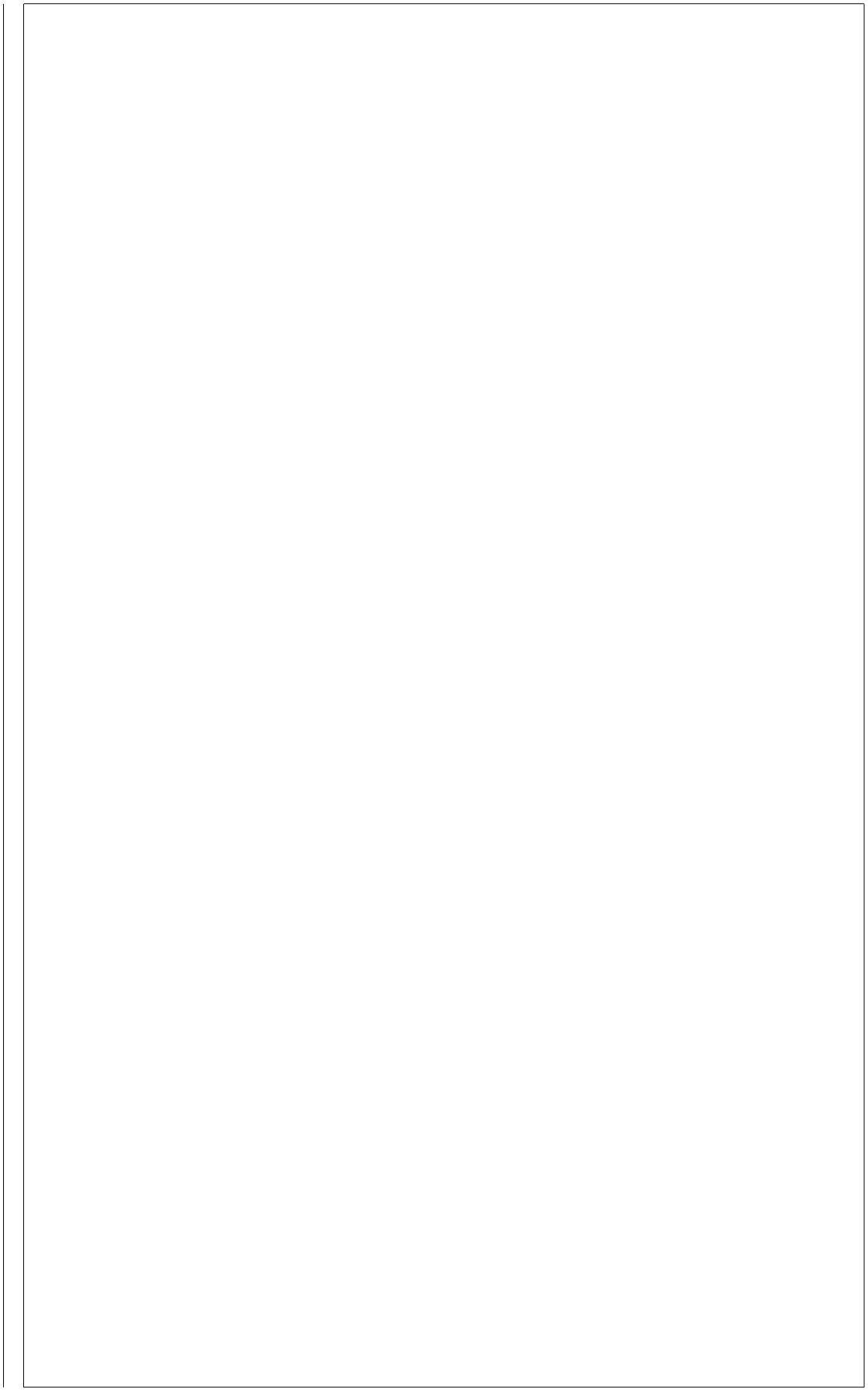
Signed 

John Gilbertson



Abstract

Abstract goes here.



Contents

Chapter 1

Introduction

Queues with scheduled arrivals occur frequently in society. These are queues where instead of customers arriving randomly, their arrival times are scheduled in advance. A common example is a doctor's surgery where patients are given appointment times. These queues also occur in shipping when ship docking times are scheduled in advance (Wang, 1997).

In this thesis, we make several assumptions about the underlying system. First, we assume a single server queue with independent exponential service times. We assume that customers arrive punctually at their scheduled arrival times. In addition, all customers have the same mean service time μ.

The objective is to find a schedule of customer arrival times that minimises a linear combination of the expected total waiting time of the customers and the expected server availability time (i.e., the expected time between the start of the first customer's service and end of the last customer's function). This linear combination is referred to as the expected cost of a schedule.

A customer's waiting time is the time from the customer's arrival until the server begins serving that customer. Throughout this thesis, we refer to the number of customers in the system at a given point in time. This number includes any customers currently being served and any customers who have arrived and are currently waiting for service.

Scheduling customers involves a trade off between minimising customer waiting time and minimising server availability time. Scheduling customers to arrive further apart decreases the expected waiting time of the customers as there are less customers in the system at the time of each customer's arrival, but increases the expected server availability time. On the other hand, scheduling customers to arrive closer together decreases the expected server availability time, but increases the expected waiting time of the customers.

Bailey (1952) was the first to study such queues. Bailey proposed a rule whereby the first two customers should be scheduled to arrive at the start of service and the remaining customers should be scheduled to arrive at fixed intervals. Bailey found that this rule reduced time wasted by customers without signficiantly increasing the server's idle time.

Many authors have extended on Bailey's ideas. Pegden and Rosenshine (1990) propose a method for determining the optimal scheduled arrival times. Mendel (2006) extends the model to allow for no-shows where a customer fails to arrive for service. Fiems, Koole, and Nain (2007) include emergency requests that immediately halt the server.

Few authors have considered the problem of adjusting a given schedule. Most authors assume that a schedule is fixed at the start of service and cannot be altered during service. This inability to alter a schedule could be a restrictive assumption. In this thesis, we examine the potential advantage of being able to adjust some of the scheduled arrival times during service.

In Chapter 3, we derive a method for determining the optimal sequence of arrival times for a given number of customers. This method assumes that the schedule is fixed for the duration of service. We call such a schedule a 'static schedule'. The results in this chapter largely follow the work of Pegden and Rosenshine (1990).

In Chapter 4, we consider a special case where customer arrivals are scheduled iteratively (i.e., one-by-one). A customer's arrival time is only decided on the arrival of the customer to be served immediately before them. Such a schedule is called a 'dynamic schedule'. This schedule is equivalent to the scheduler being able to freely adjust each customer's scheduled arrival time up until the arrival of the customer to be served immediately before them.

Towards the end of both Chapters 3 and 4, a number of example models are considered. These models help to understand the properties of each of the schedules. We examine the simple case of a small number of customers to be scheduled, and the more interesting case of 15 customers to be scheduled.

The static schedule is simply a more restrictive case of the dynamic schedule. In Chapter 5, we compare the expected cost of the two schedules. We investigate the percentage difference between each schedule's expected cost as the total number of customers.

In Chapter 6, we seek a broader understanding of the differences between the two schedules. We simulate a million runs of each schedule and compare the two cost distributions. The mean customer arrival time and customer waiting time

under each schedule is examined. This chapter gives us a fuller understanding of the properties of each schedule that are difficult to derive analytically.

Chapter 2

Literature Review

Queues with scheduled arrivals are widely studied. There is a large body of literature studying the potential of appointment systems to reduce customer waiting times and waiting room congestion. This research is essential as health care providers in particular are under a great deal of pressure to improve service quality and efficiency (Goldsmith, 1989). Before we explore the detail of this thesis, it is important to review some of the papers that study this problem.

Fomundam and Herrmann (2007), and Cayirli and Veral (2003) provide comprehensive surveys of research on appointment scheduling. Most of the papers on scheduled arrivals can be classed into two categories. Those that design algorithms to determine schedules, and those that evaluate schedules using simulation. While simulation studies can easily model complicated customer flows, queuing models often provide more generic results than simulation (Green, 2006).

The foundation paper on modeling queues with scheduled arrivals is Bailey (1952). Bailey proposes that customers' waiting times can be reduced without a significant increase in the server's idle time. The Bailey rule, which is commonly referenced in literature, is that customers should be scheduled to arrive at fixed intervals with two customers scheduled to arrive at the start of service. Bailey found that a great deal of time wasted by customers could be reduced without a signficant increase in the server's idle time. Under the Bailey rule, customers with late appointments will wait longer than those with early appointments. This lack of uniformity might be perceived as unfair and thus an undesirable property of a schedule.

Pegden and Rosenshine (1990) extend on Bailey's paper. They present an algorithm to iteratively determine the optimal arrival times for n customers that need to be scheduled. The optimal arrival times are those that minimise a weighted sum of the expected customers' waiting time and the expected server's

total availability time. Pegden and Rosenshine prove that their objective function is convex for $n \leq 4$, thus their algorithm finds the optimal schedule. While they conjecture that the objective function is convex for all n, it has not been proven.

Stein and Côté (1994) apply Pegden and Roshenshine's model to obtain numerical results for situations with more than three customers. The optimal times between successive customers become near constant as n grows. This is the often observed dome-shape. Optimal appointment intervals exhibit a common pattern where they initially increase towards the middle of a session, and then decrease. Stein and Côte simplify their model by requiring the intervals between arriving customers to be constant. This commonly studied restriction makes the model more easily applicable in practice.

Stein and Côte apply queuing theory results to solve the model for the optimal arrival interval assuming the queue reaches its steady state distribution. This assumption greatly reduces the computation required. However, in practice, it is common to find services that never reach steady state. Babes and Sarma (1991) attempted to apply steady state queuing theory, but found their results tended to be very different from those observed in real operation.

These key papers by Bailey, Pegden and Rosenshine, and Stein and Côte provide the basis for a more realistic exploration of health care systems. DeLaurentis et al. (2006) highlight that customer no-shows can lead to a waste of resources. Mendel (2006) incorporates the probability of a customer not showing up into the model presented by Pegden and Rosenshine. Unsurprisingly, no-shows lead to lower expected waiting times for customers who do show up.

The presence of walk-ins (regular and emergency) can disrupt a schedule. Gupta, Zoreda, and Kramer (1971) propose a system where non-routine requests are superimposed on top of routine scheduled requests. Fiems, Koole, and Nain (2007) investigate the effect of emergency requests on the waiting times of scheduled customers. Fiems et al. model a system with deterministic service times and discrete time. Despite this research, Cayirli and Veral suggest that walk-ins are neglected in most studies. Further research could investigate their effect on optimal arrival times.

Mondschein and Weintraub (2003) observe that the majority of the literature assumes that demand is exogenous and independent of customers' waiting times. These papers assume the total number of customers is fixed and independent of waiting times. The vast majority of servers are now private (including medical servers), so face competitive environments. Mondschein and Weintraub thus present a model where demand depends on the customers' expected waiting

time.

Few authors have attempted to model a dynamic schedule. Wang (1993) considers the problem of scheduling a new customer when there are already a number of customers with fixed scheduled arrival times. The aim is to determine the optimal arrival time for the new customer such that the objective function remains optimal. However, Wang has been criticised for not having a truly dynamic model (Erdogan and Denton, 2011). The initial scheduling of the customers ignores the possibility of an additional customer needing to be scheduled.

Simulation is a useful tool to analyse the effectiveness of appointment policies. Kao and Tung (1981) use simulation to compliment their results obtained from queuing theory. Ho and Lau (1992) study the performance of eight different appointment rules under different scenarios. They find that no rule will perform well under all circumstances.

Case studies can test the real world applications of an appointment system. While they lack generalisation, they are necessary to compliment the theoretical research. Rockart and Hofmann (1969) show individual block systems lead to more punctual doctors and patients, and less no-shows. Walter (1973) indicates that the simple grouping of inpatients and outpatients results in a substantial reduction in the doctor's idle time.

Unfortunately, Cayirli and Veral (2003) lament that despite much published work, the impact of appointment systems on outpatient clinics has been limited. Doctors are often unwilling to change their old habits. O'Keefe (1985) had their proposed appointment system of classifying patients rejected by staff. Huarng and Hou Lee (1996) were unable to implement their system due to staff resistance. Bennett and Worthington (1998) found their recommendations were not implemented successfully. Future research should attempt to develop models that will be accepted and implemented in real health care services.

Chapter 3

Static Schedules

This chapter largely follows the results of Pegden and Rosenshine (1990). The aim is to derive a method for choosing an optimal static schedule. A static schedule is a sequence of n customer arrival times $t_1, \ldots, t_n$ chosen at the start of service and fixed for the duration of service.

For simplicity, these results assume the customer service times are independent and identically distributed (iid) exponential random variables with mean μ. There is a single server. All customers are punctual and arrive at their scheduled arrival time.

3.1 Objective Function

The optimal schedule minimises the expected cost, which is a linear combination of the expected total customers' waiting time and the expected server availability time.

Denote the expected waiting time of customer i as w_i. The expected total customers' waiting time is the sum of the individual customer's expected waiting times,

$$\mathbb{E}\Big[\text{total customer's waiting time}\Big] = \sum_{i=1}^{n} w_i. \tag{3.1}$$

Instead of solving for the customer arrival times, it is easier to solve for the arrival time of the first customer t_1 and the customer interarrival times $\mathbf{x}_n = (x_1, \ldots, x_{n-1})$ where $x_i = t_{i+1} - t_i$. The expected server availability time is the expected time from the start of service until the end of service. This time is the summation of the last customer's scheduled arrival time, the last customer's

expected waiting time and the last customer's expected service time,

$$\mathbb{E}\Big[\text{server availability time}\Big] = \left(t_1 + \sum_{i=1}^{n-1} x_i\right) + w_n + \mu. \tag{3.2}$$

Denote c_W and c_S as the per unit time cost of the expected total customer's waiting time and the per unit time cost of the expected server availability time respectively. The objective function to be minimised is thus,

$$\phi(t_1, \mathbf{x}_n) = c_W \sum_{i=1}^{n} w_i + c_S \left[t_1 + \sum_{i=1}^{n-1} x_i + w_n + \mu\right]. \tag{3.3}$$

The first customer should obviously be scheduled for the start of service so $t_1 = 0$. Moreover, can scale the objective function by dividing by $(c_W + c_S)$ and defining $\gamma = \frac{c_S}{c_W + c_S}$,

$$\phi(\mathbf{x}_n) \coloneqq (1-\gamma) \sum_{i=1}^{n} w_i + \gamma \left[\sum_{i=1}^{n-1} x_i + w_n + \mu\right]. \tag{3.4}$$

The optimal static schedule is thus the interarrival times that minimise Equation 3.4

3.2 Expected Waiting Time

We want to express w_i (i.e., the expected waiting time of customer i) as a function of the interarrival times $\mathbf{x}_n$. If there are j customers in the system immediately prior to the arrival of customer i, then $w_i = j\mu$ by the memoryless property of the exponential distribution. The number of customers in the system refers to both customers being served and customers waiting for service.

Denote the number of customers in the system immediately prior to the arrival of customer i by N_i. Thus, the expected waiting time of customer i is given by

$$w_i = \sum_{j=0}^{i-1} (j\mu)\mathbb{P}(N_i = j). \tag{3.5}$$

It is more common in queuing theory to consider the queue length as a right-continuous process. This would involve considering the number of customers in the system on the arrival of the customer i. However, in this chapter, we choose to remain consistent with the notation of Pegden and Rosenshine (1990) and consider the system size as a left-continuous process.

Pegden and Rosenshine (1990) derive the following complicated recursive equation for calculating the probability of a given number of customers in the system,

$$\mathbb{P}(N_i = j) = \begin{cases} 1 & \text{for } i = 1,\ j = 0 \\ \sum_{k=1}^{i-1} \mathbb{P}(N_{i-1} = k-1) \left[1 - \sum_{l=0}^{k-1} \frac{x_{i-1}^l}{\mu^l l!} \mathrm{e}^{\frac{-x_{i-1}}{\mu}}\right] & \text{for } i \geq 2,\ j = 0 \\ \sum_{k=0}^{i-j-1} \mathbb{P}(N_{i-1} = j+k-1) \left[\frac{x_{i-1}^k}{\mu^k k!} \mathrm{e}^{\frac{-x_{i-1}}{\mu}}\right] & \text{for } i \geq 2,\ j \geq 1. \end{cases} \tag{3.6}$$

Pegden and Rosenshine's formulation can be solved more efficiently by identifying that $\{N_i\}$ forms a discrete time Markov chain with nonstationary transition probabilities. Stein and Côté (1994) show that the transition from N_{i-1} to N_i is governed by the transition matrix

$$T(x_{i-1}) = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & \ldots & n-1 & n \end{array} \\ \begin{pmatrix} r_0 & a_0 & 0 & 0 & \ldots & 0 & 0 \\ r_1 & a_1 & a_0 & 0 & \ldots & 0 & 0 \\ r_2 & a_2 & a_1 & a_0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ r_{n-1} & a_{n-1} & a_{n-2} & a_{n-3} & \ldots & a_1 & a_0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \end{array},$$

where

$$a_k = \frac{x_{i-1}^k}{\mu^k k!} \mathrm{e}^{\frac{-x_{i-1}}{\mu}}, \tag{3.7}$$

and

$$r_k = 1 - \sum_{l=0}^{k} a_l. \tag{3.8}$$

The last row of $T(x_{i-1})$ is arbitrary as it is not possible to have n customers in the system immediately before a customer's arrival.

Instead of Equation 3.6, for $i \geq 2$, the distribution of N_i is more clearly expressed by

$$\begin{aligned} & \begin{pmatrix} \mathbb{P}(N_i = 0) & \ldots & \mathbb{P}(N_i = n) \end{pmatrix} \\ = & \begin{pmatrix} \mathbb{P}(N_{i-1} = 0) & \ldots & \mathbb{P}(N_{i-1} = n) \end{pmatrix} T(x_{i-1}) \\ = & \begin{pmatrix} \mathbb{P}(N_1 = 0) & \ldots & \mathbb{P}(N_1 = n) \end{pmatrix} T(x_1) T(x_2) \cdots T(x_{i-1}). \end{aligned} \tag{3.9}$$

The distribution of N_1 (i.e., the initial distribution) is given by

$$\begin{pmatrix} \mathbb{P}(N_1 = 0) & \dots & \mathbb{P}(N_1 = n) \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}. \tag{3.10}$$

Therefore, the expected waiting time of customer i can be compactly expressed using the transition matrices as

$$w_i = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} T(x_1)T(x_2)\cdots T(x_{i-1}) \begin{pmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ n \end{pmatrix}. \tag{3.11}$$

3.3 Example Models

3.3.1 Model for 2 Customers

We consider the simplest case of this model where there are two customers to be scheduled (i.e., $n = 2$). As the first customer is scheduled to arrive at the start of service, the only unknown variable is the optimal interarrival time between the first and second customer (i.e., x_1).

By Equation 3.11, the expected waiting times of the two customers are

$$w_1 = 0, \tag{3.12}$$

and

$$w_2 = \mu \mathrm{e}^{\frac{-x_1}{\mu}}. \tag{3.13}$$

By Equation 3.4, the objective function to be minimised is given by

$$\phi(x_1) = \mu \left[\gamma + \exp\left(\frac{-x_1}{\mu} \right) \right] + \gamma x_1. \tag{3.14}$$

This objective function is convex as

$$\forall x_1 \quad \phi''(x_1) = \frac{1}{\mu} \exp\left(\frac{-x_1}{\mu} \right) > 0. \tag{3.15}$$

Due to the convexity of the objective function, the optimal policy that min-

mises $\phi(x_1)$ can be found by solving:

$$\phi'(x_1) = 0 \implies -\exp\left(\frac{-x_1}{\mu}\right) + \gamma = 0. \tag{3.16}$$

Therefore, the optimal policy is

$$x_1^* = \arg\min_{x_1 \geq 0} \phi(x_1) = -\mu \ln \gamma. \tag{3.17}$$

As the server availability cost increases relative to the customer waiting cost (i.e., γ increases), the second customer is scheduled to arrive earlier (i.e., x_1^* decreases).

Unfortunately, as Pegden and Rosenshine (1990) found, no general algebraic solution exists for more than two customers (i.e., $n \geq 3$). All other cases need to be solve numerically. Pegden and Rosenshine prove that the objective function $\phi(\mathbf{x}_n)$ is convex for $n = 1, 2, 3, 4$. Moreover, they conjecture that it appears to be convex for general n, but are unable to prove it.

We developed an algorithm in Python to efficiently find a numeric approximation to the optimal static schedule. The algorithm is included in the appendix. We found that the algorithm can find the schedule when scheduling up to 15 customers. Our algorithm uses the 'L-BFGS-B' method in `scipy.optimize.minimize`. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is an iterative method for solving an unconstrained optimization problem. The 'L-BFGS-B' is an extension of the BFGS algorithm to handle simple bound constraints on the variables. For a detailed explanation of the 'L-BFGS-B' method, see Byrd et al. (1995).

3.3.2 Model for 15 Customers

We consider here the case of $n = 15$. Without loss of generality, can let $\mu = 1$. For $\mu \neq 1$, the solutions found are the optimal values of $\mu\mathbf{x}_n = (\mu x_1, \ldots, \mu x_{n-1})$.

Figure 3.1 plots the optimal interarrival times $\mathbf{x}_n^* = (x_1^*, \ldots, x_{14}^*)$ found by our algorithm for scheduling 15 customers with a mean service time $\mu = 1$ for various values of γ.

The optimal interarrival times increase for the initial few customers, remain constant for the majority of customers, and then decrease for the last few customers. This is the dome-shape that was observed by Stein and Côté (1994) and Mendel (2006). The observation that the first customers arrive close together obeys Bailey's Low that was first recommended by Bailey (1952). As γ increases (i.e., the relative cost of server availability), the optimal interarrival

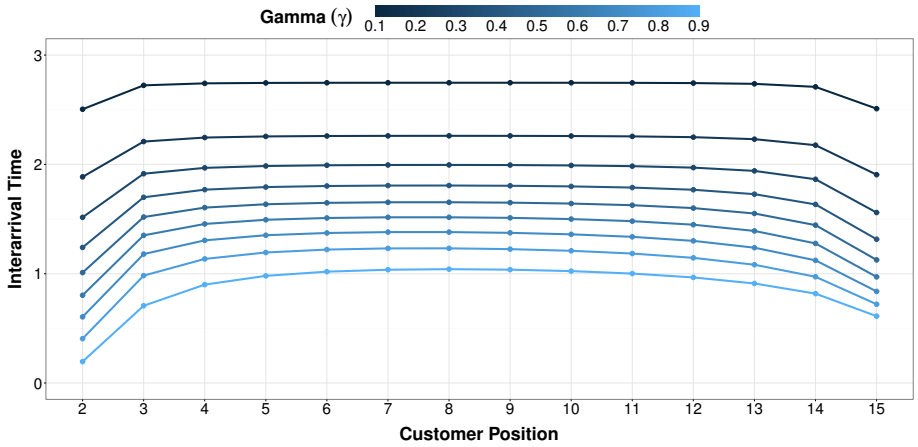


Figure 3.1: Optimal interarrival time for each of the 14 customers after the first customer. Figure generated assuming $\mu = 1$. Each line is plotted for a fixed value of $\gamma = \frac{c_S}{c_S + c_W}$. The darkest line is for $\gamma = 0.1$. As γ increases, the lines become lighter.

times decrease while obeying the same general shape.

The common approach to efficiently finding the optimal static schedule involves simplifying the model by assuming a constant interarrival time (i.e., $x_1 = \ldots = x_{n-1}$). Stein and Côté (1994) justify this simplification by claiming that for any single server queue with exponential service times, the average waiting time of a customer will be at a minimum for constant interarrival times. If this justification holds true, then this guarantees that constant interarrival times is optimal for $\gamma = 0$ (i.e., $c_S = 0$). However, it does not imply that it's optimal for $\gamma > 0$.

The constant interarrival simplification significantly reduces computation cost, but Figure 3.1 suggests that it is clearly not optimal for the first few and last few customers. In addition, it is not possible to include this simplification in the dynamic schedule model presented in the following chapter. Thus, this simplification is not pursued here.

Chapter 4

Dynamic Schedules

The static schedule presented in Chapter 3 and applied commonly in the scheduled arrivals literature is fixed for the duration of service. It would intuitively be advantageous to allow the schedule to vary during service. For example, if the service times of the first few customers are longer than expected, then it would be beneficial to schedule the remaining customers to arrive later.

This chapter approaches the problem of choosing a schedule in a similar way to Chapter 3. The aim is to find the arrival time of the first customer and the customer interarrival times that minimise the expected cost. The assumptions of iid service times and punctual customers are the same as Chapter 3.

The dynamic schedule is chosen progressively during service. Immediately after a customer arrives and begins waiting for service, the scheduler chooses the arrival time of the next customer. Most real-world situations are obviously more restrictive than this. It is often not possible to schedule the customer arrivals one-by-one.

Real-world situations may sometimes allow a degree of flexibility in regards to rescheduling customers. The theory is that if the dynamic schedule presented here is significantly better than the static schedule, than (if possible) the rescheduling ability should be included in real-world models for scheduled arrivals. However, if the dynamic schedule performs similarly to the static schedule than it is likely reasonable to ignore the rescheduling in the model formulation.

4.1 Markov Decision Process

We consider the problem of scheduling M customers. Denote the number of customers in the system on arrival of customer i by k_i. In a similar way to the sequence $\{N_i\}$ in Chapter 3, the sequence $\{k_1, \ldots, k_M\}$ is a discrete time

Markov chain with a finite horizon. In contrast to Chapter 3, we follow the general convention here and define $\{k_i\}$ to be a right-continuous process with left limits.

We denote the states of the Markov chain as (n, k). This state refers to n customers remaining to be scheduled and k customers currently in the system (i.e., either waiting or being served). The state variable n is treated as the current 'time' of the Markov chain. On arrival of the first customer, the system is in state $(M-1, k_1)$. On arrival of the second customer, the system is in state $(M-2, k_2)$. On arrival of the last customer, the system is in state $(0, k_M)$.

For $n \geq 1$, at each state (n, k) the scheduler needs to schedule the arrival of the next customer relative to the current time. We denote this time (i.e., action) by a. The scheduler cannot schedule the customer to arrive in the past so the set of possible actions is $A = [0, \infty)$. The time a must be chosen such that $a \in A$.

This process is naturally modelled as a Markov decision process. The aim is to find a policy for the scheduler to choose the optimal value of a. Denote the optimal policy by a^*.

4.2 Expected Cost of Schedule

As in Chapter 3, the expected cost of a schedule is a linear combination of the expected total customers' waiting time and the expected server availability time. Denote the expected cost of the state (n, k) given that the next customer is scheduled to arrive in a time units by $C_n(a, k)$. The optimal policy minimises the expected cost of the state (n, k). Denote this minimum by $C_n^*(k)$ such that

$$C_n^*(k) = C_n(a^*, k) = \min_{a \geq 0} C_n(a, k). \tag{4.1}$$

Each transition in the Markov chain decrements the value of n and transitions to a new value of k. Suppose the next customer is scheduled to arrive in a time units. Consider the transition from state (n, k) to state $(n-1, j)$ over time interval a. Denote the probability of this transition by $p_a(k, j)$, and the expected cost during this transition by $R_a(k, j)$. Note that j is the total number of customers in the system immediately after the next customer's arrival.

Bellman's Principle of Optimality (Sniedovich, 1986) states that

> An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an

optimal policy with regard to the state resulting from the first decision.

This implies that the expected cost of the current state (under the optimal policy) is a function of the expected cost involved in transitioning to the next state, the expected cost of the next state (under the optimal policy) and the probability of transitioning to the next state over all possible next states.

For $n \geq 1$, the expected cost of state (n, k) can be computed by the Bellman equation (Bellman, 1957),

$$C_n^*(k) \coloneqq \min_{a \geq 0} C_n(a, k) = \min_{a \geq 0} \left[\sum_{j=1}^{k+1} p_a(k, j) \Big(R_a(k, j) + C_{n-1}^*(j) \Big) \right]. \tag{4.2}$$

Equation 4.2 is a recursive equation involving C^*. The optimal dynamic schedule is found by solving for each optimal interarrival time a iteratively.

4.3 Base Case

Solving Equation 4.2 requires a solution for the base case where $n = 0$. The state $(0, k)$ is the state where there are no customers remaining to be scheduled and k customers currently in the system. No futher scheduling decisions are required at this state.

Denote the expected waiting time of the customer that is currently in position i as w_i. This expected waiting time is the summation of the expected service times of all the customers in positions $\{1, \ldots, i-1\}$. The expected total customers' waiting time for the k remaining customers is the sum of their individual waiting times,

$$\begin{aligned} \mathbb{E}\Big[\text{total customer's waiting time at state } (0, k)\Big] &= \sum_{i=1}^{k} w_i \\ &= \sum_{i=1}^{k} \mu(i-1) \\ &= \frac{\mu k(k-1)}{2}. \end{aligned} \tag{4.3}$$

If there are no customers remaining to be scheduled, then the expected server availability time is the expected time until the customer currently in the last position in the queue finishes service. This time is the summation of the expected

service times of the k customers currently in the system,

$$\mathbb{E}\Big[\text{server availability time at state } (0,k)\Big] = \sum_{i=1}^{k} \mu = k\mu. \tag{4.4}$$

The per unit time costs c_W and c_S are defined as before in Chapter 3. The expected cost of the base case is thus given by

$$C_0^*(k) = c_W \frac{\mu k(k-1)}{2} + c_S k\mu. \tag{4.5}$$

In order to compare $C_0^*(k)$ with the cost of the static schedule, need to scale it by dividing by $(c_S + c_W)$ and defining $\gamma = \frac{c_S}{c_S + c_W}$,

$$C_0^*(k) \coloneqq (1-\gamma)\frac{\mu k(k-1)}{2} + \gamma k\mu. \tag{4.6}$$

4.4 Erlang Distribution

The transition probability and expected transition cost for the dynamic schedule depend on the distribution function and conditional expectations of the Erlang distribution, which are shown here.

Denote the service time of the customer that is currently in position i in the queue as S_i. For n customers, the service times $S_1, \ldots, S_n$ are iid exponential random variables with mean μ.

For $r \geq 1$, the waiting time of the customer in position $(r+1)$ is given by

$$X = \sum_{i=1}^{r} S_i \sim \text{Erlang}(r, \mu), \tag{4.7}$$

which has density

$$f(x; r) \coloneqq \frac{1}{\mu(r-1)!} \left(\frac{x}{\mu}\right)^{r-1} \mathrm{e}^{\frac{-x}{\mu}}, \tag{4.8}$$

and distribution function

$$F(x; r) \coloneqq \mathbb{P}(X \leq x) = \begin{cases} 0 & \text{for } x = 0 \\ 1 - \sum_{i=0}^{r-1} \frac{1}{i!} \left(\frac{x}{\mu}\right)^i \mathrm{e}^{\frac{-x}{\mu}} & \text{for } x > 0. \end{cases} \tag{4.9}$$

For $x \geq 0$, $F(x;r)$ is continuous as

$$\lim_{x \to 0^+} F(x;r) = 0 = F(0;r). \tag{4.10}$$

Computing the expected transition cost requires the conditional expectation of X given $X \leq a$. We are unable to find reference to this conditional expectation in the literature so derive it here (for $a > 0$),

$$\begin{aligned}
\mathbb{E}[X|X \leq a] &= \int x \ \mathbb{P}(X \in dx | X \leq a) \\
&= \frac{1}{\mathbb{P}(X \leq a)} \int x \ \mathbb{P}(X \in dx, X \leq a) \\
&= \frac{1}{F(a;r)} \int_0^a x f(x;r) dx \\
&= \frac{\mu r}{F(a;r)} \int_0^a \frac{1}{\mu r!} \left(\frac{x}{\mu}\right)^r \mathrm{e}^{\frac{-x}{\mu}} dx \\
&= \frac{\mu r F(a;r+1)}{F(a;r)}.
\end{aligned} \tag{4.11}$$

Moreover, $\mathbb{E}[X|X \leq 0] = 0$ by definition. Therefore, the conditional expectation of X given $X \leq a$ is defined as

$$G(a;r) \coloneqq \mathbb{E}[X|X \leq a] = \begin{cases} 0 & \text{for } a = 0 \\ \frac{\mu r F(a;r+1)}{F(a;r)} & \text{for } a > 0. \end{cases} \tag{4.12}$$

This expression makes intutive sense. For $a > 0$ and $r \geq 1$, $\frac{F(a;r+1)}{F(a;r)} < 1$. In addition, the mean of the Erlang distribution is $\mathbb{E}[X] = \mu r$. Thus, for $a > 0$ and $r \geq 1$, $\mathbb{E}[X|X \leq a] < \mathbb{E}[X]$ as expected.

Moreover,

$$\lim_{a \to \infty} \frac{F(a;r+1)}{F(a;r)} = \frac{\lim_{a \to \infty} F(a;r+1)}{\lim_{a \to \infty} F(a;r)} = \frac{1}{1} = 1. \tag{4.13}$$

Thus, $\lim_{a \to \infty} \mathbb{E}[X|X \leq a] = \mathbb{E}[X]$ as expected.

Finally, suppose there is an exponential random variable Y with mean μ that is independent of X. The expected transition cost also requires the conditional expectation of X given $X \leq a$ and $X + Y > a$. We were again unable to find reference to this conditional expectation in the literature so derive it here (for

$a > 0$),

$$\begin{aligned}
&\mathbb{E}[X|X \le a, X+Y > a] \\
&= \int x\ \mathbb{P}(X \in dx | X \le a, X+Y > a) \\
&= \frac{1}{\mathbb{P}(X \le a, X+Y > a)} \int x\ \mathbb{P}(X \in dx, X \le a, X+Y > a) \\
&= \frac{1}{\mathbb{P}(X \le a, X+Y > a)} \iint_D x\ \mathbb{P}(X \in dx, X \le a, X+y > a) f(y;1) dy.
\end{aligned} \tag{4.14}$$

The double integral needs to be evaluated over

$$D = \Big\{(x,y) : x \ge 0, y \ge 0, x \le a, x+y > a\Big\}. \tag{4.15}$$

D can be expressed as the union of two disjoint sets in the following way,

$$D = \Big\{(x,y) : x \in (a-y, a], y \in [0,a]\Big\} \bigcup \Big\{(x,y) : x \in [0,a], y \in (a,\infty)\Big\}. \tag{4.16}$$

In addition, we show in the appendix that

$$\mathbb{P}(X \le a, X+Y > a) = \frac{1}{r!}\left(\frac{a}{\mu}\right)^r \mathrm{e}^{\frac{-a}{\mu}}. \tag{4.17}$$

Thus, the conditional expectation can be further derived below,

$$\begin{aligned}
&\mathbb{E}[X|X \le a, X+Y > a] \\
&= \frac{1}{\frac{1}{r!}\left(\frac{a}{\mu}\right)^r \mathrm{e}^{\frac{-a}{\mu}}} \left[\int_0^a \int_{a-y}^a + \int_a^\infty \int_0^a\right] x f(x;r) f(y;1) dx dy \\
&= \frac{\frac{a}{r+1}\frac{1}{(r-1)!}\left(\frac{a}{\mu}\right)^r \mathrm{e}^{\frac{-a}{\mu}}}{\frac{1}{r!}\left(\frac{a}{\mu}\right)^r \mathrm{e}^{\frac{-a}{\mu}}} \\
&= \frac{ar}{r+1}.
\end{aligned} \tag{4.18}$$

Moreover, $\mathbb{E}[X|X \le 0, X+Y > 0] = 0$ by definition. Therefore, the conditional expectation of X given $X \le a$ and $X+Y > a$ is defined as

$$H(a;r) \coloneqq \mathbb{E}[X|X \le a, X+Y > a] = \frac{ar}{r+1}. \tag{4.19}$$

Perhaps surprisingly, the conditional expectation $H(a;r)$ does not depend on the parameter μ.

4.5 Transition Probability

The transition probability $p_a(k, j)$ is the probability that the number of customers in the system changes from k customers initially to j customers on the arrival of the next customer after a time units. In other words, it is the probability that there are $k - (j - 1)$ departures from the queue over a time interval of length a.

The transition probability is most easily derived on a case by case basis. The full derivation is included in the appendix and only the final equation is presented here,

$$p_a(k,j) \coloneqq \begin{cases} \mathbb{1}(j=1) & \text{for } k = 0 \\ F(a;k) & \text{for } k \geq 1, j = 1 \\ F(a;k-j+1) - F(a;k-j+2) & \text{for } k \geq 2, 2 \leq j \leq k \\ 1 - F(a;1) & \text{for } k \geq 1, j = (k+1) \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

Given a current state (n, k) and a time interval a, the total probability over all possible next states $\{(n-1, j) : 1 \leq j \leq k+1\}$ is

$$\sum_{j=1}^{k+1} p_a(k,j) = 1. \tag{4.21}$$

4.6 Expected Transition Cost

The expected transition cost is the expected cost of transitioning from state (n, k) to state $(n-1, j)$ if the next customer is scheduled to arrive in a time units.

In a similar way to the transition probability, the expected transition cost is derived on a case by case basis. To compare the dynamic schedule with the static schedule, the cost is scaled by dividing by $(c_S + c_W)$ and defining $\gamma = \frac{c_S}{c_S + c_W}$. The full derivation is included in the appendix and the final equation is presented here,

$$R_a(k,j) \coloneqq \begin{cases} \gamma a & \text{for } k \in \{0,1\} \\ (1-\gamma)\frac{G(a;k)(k-1)}{2} + \gamma a & \text{for } k \geq 2, j = 1 \\ (1-\gamma)\frac{a(k+j-3)}{2} + \gamma a & \text{for } k \geq 2, 2 \leq j \leq k+1. \end{cases} \tag{4.22}$$

4.7 Example Models

4.7.1 Model for 2 Customers

Assume there are two customers that need to be scheduled for service. The initial state is $(2, 0)$ (i.e., two customers to schedule and no customers currently in the system). The possible states during service are all the states in the set

$$\Big\{(n, k) \in \{0, 1, 2\}^2 : n + k \leq 2\Big\}. \tag{4.23}$$

.

From Equation 4.2, for $n \geq 1$, the expected cost of state $(n, 0)$ (i.e., no customers currently in the system) as a function of the time interval a until the next customer arrival is given by

$$C_n(a, 0) = C^*_{n-1}(1) + \gamma a. \tag{4.24}$$

As $\gamma \in (0, 1)$, the optimal policy a^* where $C_n(a, 0)$ attains a minimum is

$$a^* = \arg\min_{a \geq 0} C_n(a, 0) = 0. \tag{4.25}$$

Thus, if there are no customers currently waiting, then the next customer should be scheduled to arrive immediately. This agrees with the result of the static schedule that the first customer should be schedule to arrive immediately at the state of service.

As the transition occurs immediately, the expected cost of state $(n, 0)$ equals the expected cost of state $(n - 1, 1)$ for $n \geq 1$,

$$C^*_n(0) = C^*_{n-1}(1). \tag{4.26}$$

From Equation 4.2, for $n \geq 1$, the expected cost of state $(n, 1)$ (i.e., one customer currently in the system) as a function of the time interval a until the next customer arrives is given by

$$C_n(a, 1) = C^*_{n-1}(1) + \mathrm{e}^{\frac{-a}{\mu}} \Big[C^*_{n-1}(2) - C^*_{n-1}(1)\Big] + \gamma a. \tag{4.27}$$

The optimal policy a^* where $C_n(a, 1)$ attains a minimum is

$$a^* = \arg\min_{a \geq 0} C_n(a, 0) = \mu \ln \left[\frac{C^*_{n-1}(2) - C^*_{n-1}(1)}{\gamma\mu}\right]. \tag{4.28}$$

Returning to the case of scheduling two customers using this dynamic schedule. By Equation 4.25, the first customer should be scheduled to arrive immediately. On the first customer's arrival, the system is at state $(1, 1)$. By Equation 4.28, the second customer should be scheduled to arrive a^* after the arrival of the first customer where

$$a^* = \mu \left[\frac{C_0^*(2) - C_0^*(1)}{\gamma\mu} \right] = -\mu \ln \gamma. \tag{4.29}$$

This dynamic schedule is exactly the same as the static schedule for two customers. As the first customer is scheduled to arrive at the start of service, all arrival times are decided at the start of service. Therefore, the optimal policy is the same for both schedules.

Unfortunately, for $k \geq 2$, no algebraic solution exists for the optimal policy for state (n, k). Thus, we cannot analytically derive the dynamic schedule for more than two customers. Those cases need to be solved numerically.

To further illustrate this point, we consider the cost of the state $(1, 2)$ (i.e., one customer remaining to schedule and two customers currently in the system). From Equation 4.2, the expected cost of this state as a function of the time until the last customer's arrival (a) is given by

$$C_1(a, 2) = \gamma a + \mu + \mathrm{e}^{\frac{-a}{\mu}}(a + 2\mu). \tag{4.30}$$

The optimal policy a^* where $C_1(a, 2)$ attains a minimum is given by the a^* that solves the equation

$$-\left(1 + \frac{a^*}{\mu}\right) \mathrm{e}^{-\left(1+\frac{a^*}{\mu}\right)} = -\gamma \mathrm{e}^{-1}. \tag{4.31}$$

The optimal policy can be defined in terms of the Lambert W function as

$$a^* = -\mu \left[1 + W\left(-\gamma \mathrm{e}^{-1}\right)\right]. \tag{4.32}$$

The Lambert W function is the inverse function of

$$f(W) = W\mathrm{e}^W. \tag{4.33}$$

Unfortunately, W is not an elementary function (Chow, 1999). The optimal arrival time of the last customer needs to be computed numerically for a given value of γ.

4.7.2 Model for 15 Customers

We consider here the case of $n = 15$. Without loss of generality, can let $\mu = 1$. For $\mu \neq 1$, the optimal policies a^* found are the optimal interarrival times μa^*. For simplicity, assume that the per unit time costs c_W and c_S are both equal such that $\gamma = 0.5$.

In a similar way to Chapter 3, we developed an algorithm in Python to efficiently find a numeric approximation to the optimal dynamic schedule. The algorithm is included in the appendix. We again use the 'L-BFGS-B' method of `scipy.optimize.minimize` in Python. Finding the optimal dynamic schedule for 15 customers involves solving several contrained optimisation problems. We solve for the optimal policy a^* at each possible state in the set

$$\Big\{(n,k) \in \{0,\dots,15\}^2 : n + k \leq 15\Big\}. \tag{4.34}$$

Our algorithm finds the optimal policy at each of the states in the order given by

$$(0,0),(0,1),\dots,(0,15),(1,0),\dots,(1,14),(2,1),\dots,(14,1),(15,0). \tag{4.35}$$

By choosing this order and storing the results of previous computations, our algorithm is able to easily find the dynamic schedule for 15 customers. This algorithm is significantly more efficient than our algorithm for finding the optimal static schedule. Moreover, due to the nature of the dynamic schedule, the optimal schedule for scheduling $n = 15$ customers includes the optimal schedule for scheduling $n \leq 15$ customers.

We computed the expected cost of each state (n, k) using our algorithm and the results are plotted in Figure 4.1.

The cost of the initial state $(15, 0)$ is 13.55, thus the expected cost of serving 15 customers with a dynamic schedule is 13.55. The worst state in Figure 4.1 is the state $(0, 15)$, which is all 15 customers in the system. This can occur if all 15 customers are scheduled to arrive immediately at the start of service (to ensure minimal server availability time) or if the first customer has an extremely long service time.

The lines plotted on Figure 4.1 are the expected costs with fixed n and varying k. As the number of customers in the system (k) increases, the expected cost increases exponentially. By contrast, as the number of customers to be scheduled (n) increases with k fixed, the expected cost of the state increases approximately linearly at a significantly smaller rate.

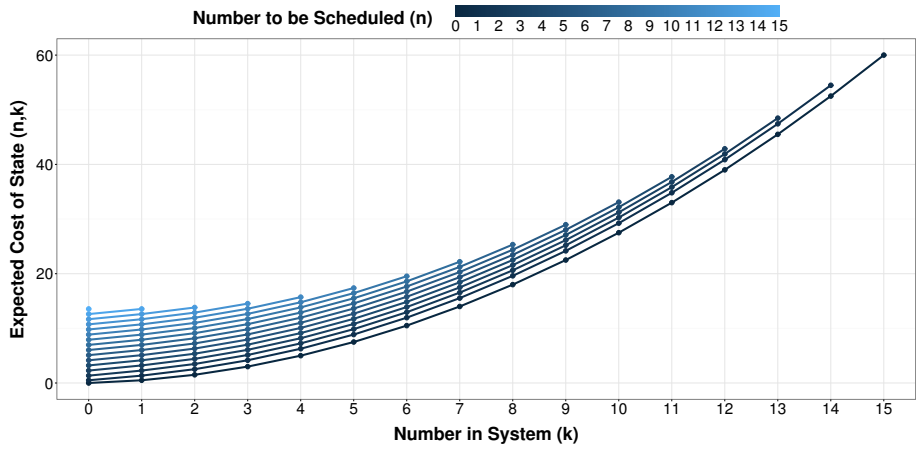


Figure 4.1: Expected cost of possible states with 15 total customers. Figure generated assuming $\gamma = \frac{c_S}{c_S+c_W} = 0.5$ and $\mu = 1$. Each line is plotted for a fixed value of the number of customers still to be scheduled (n). The darkest line is for $n = 0$. As n increases, the lines become lighter.

It's difficult to observe, but for $n \geq 1$, Figure 4.1 shows that $C_n^*(0) = C_{n-1}^*(1)$. This confirms the previous result that if there are no customers currently in the system (e.g., initially), then it is always optimal to schedule the next arrival immediately. The expected cost does not change as the next arrival occurs immediately.

Figure 4.2 plots the corresponding interarrival times a^* for the states plotted in Figure 4.1. This figure does not include optimal times for the states with $n = 0$ as there is no next customer to schedule in those states.

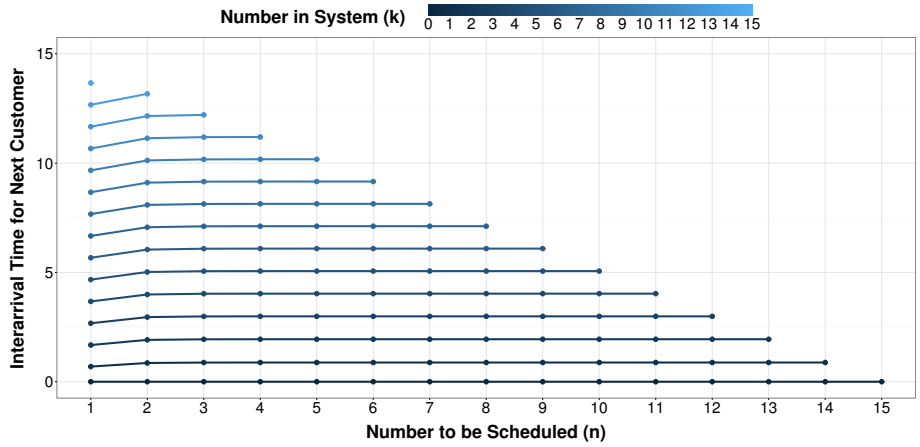


Figure 4.2: Optimal interarrival time for each possible state with 15 total customers. Figure generated assuming $\gamma = \frac{c_S}{c_S+c_W} = 0.5$ and $\mu = 1$. Each line is plotted for a fixed value of the number of customers in the system (k). The darkest line is for $k = 0$. As k increases, the lines become lighter.

Understanding Figure 4.2 is helped by looking at some examples. The optimal interarrival time for the initial state $(15, 0)$ is 0. The first customer should be scheduled to arrive immediately. The optimal interarrival time for the state $(3, 10)$ is 10.17. If (on arrival of a customer), there are 10 customers in the system and 3 customers remaining to be scheduled, then the next customer should be scheduled to arrive in 10.17 time units. This is slightly greater than the expected service time of the 10 customers currently in the system to account for the customer waiting cost.

The first pattern to notice is that if there are no customers currently waiting (i.e., $k = 0$), then (as discussed earlier) the optimal policy is to schedule the next arrival immediately. This makes intutive sense as scheduling the next arrival immediately minimises the expected server availability time without effecting the expected total customers' waiting times.

As k increases for fixed n, the optimal interarrival times a^* appear to increase at a slightly decreasing rate. The optimal a^* increases by approximately μ for each additional k. In contrast, for $n \geq 2$ and fixed k, a^* appears to be constant at a value close to μk (i.e., the expected time for the system to be empty). The optimal a^* appears to be independent of the the number of customers still to be scheduled provided that there are at least two customers still to be scheduled.

Chapter 5

Schedule Comparison

5.1 Expected Cost Comparison

Suppose that there are N customers to be scheduled. In the case of the static schedule, the optimal policy is given by $\mathbf{x}_N^* = (x_1^*, \ldots, x_{N-1}^*)$ and the expected cost of the optimal policy is given by $\phi(\mathbf{x}_N^*)$. In the case of the dynamic schedule, the initial state is $(N, 0)$, so the expected cost of the optimal policy is given by $C_N^*(0)$.

Clearly, as the dynamic schedule has the ability to match the optimal policy for the static schedule, $C_N^*(0) \leq \phi(\mathbf{x}_N^*)$ (i.e., the dynamic schedule cannot have greater expected cost) regardless of the number of customers to be scheduled. As found in Chapter 4, equality is attained for the cases where $N \in \{0, 1, 2\}$ as the schedules are identical in those cases.

Figure 5.1 plots the expected costs of both schedules against the number of customers to be scheduled (N) assuming $\gamma = 0.5$ and $\mu = 1$.

As expected, the costs are identical for $N \in \{0, 1, 2\}$. For all other N values, the cost of the static schedule is greater than the cost of the dynamic schedule. The cost difference appears to be minimal for $N \leq 5$, but as N increases the cost difference increases.

5.2 Expected Percentage Cost Saving

Define the expected percentage cost saving ΔC (i.e., the percentage difference between the expected cost of the static schedule and the dynamic schedule) as

$$\Delta C := 100 \times \frac{\phi(\mathbf{x}_N^*) - C_N^*(0)}{\phi(\mathbf{x}_N^*)}. \tag{5.1}$$

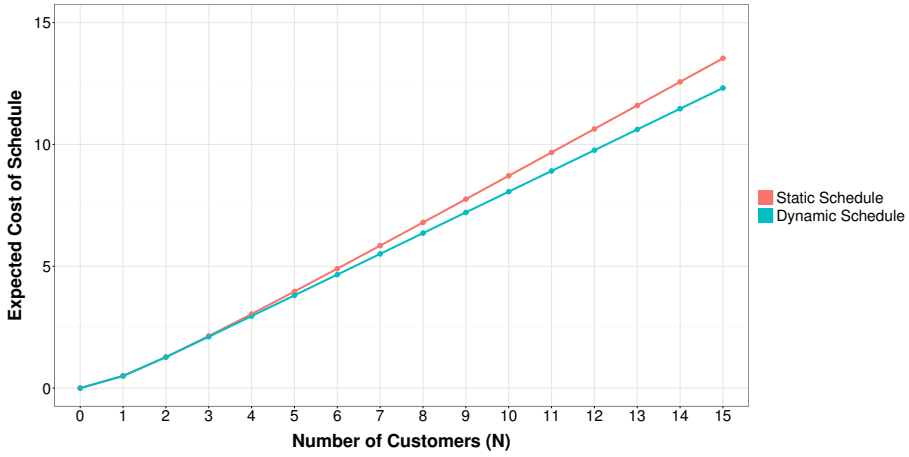


Figure 5.1: Plot of the expected cost of each schedule against the number of customers to be scheduled (N) for both the static and dynamic schedules where $N \in \{0, \ldots, 15\}$, $\gamma = \frac{c_S}{c_S + c_W} = 0.5$ and $\mu = 1$.

Figure 5.2 plots the percentage cost saving (ΔC) against γ for various values of the number of customers (N) to be scheduled.

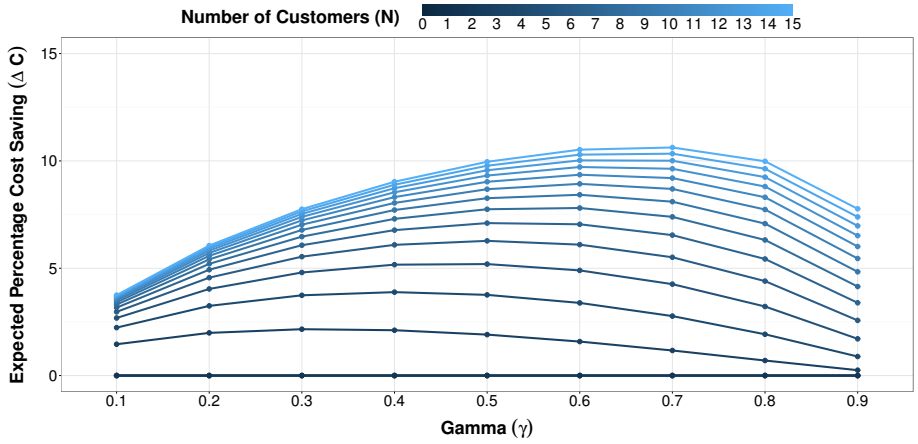


Figure 5.2: Plot of the percentage cost saving (ΔC) against $\gamma = \frac{c_S}{c_S + c_W}$ where $N = \{0, \ldots, 15\}$ and $\mu = 1$.

For $N \in \{0, 1, 2\}$, $\Delta C = 0$ for all values of γ as the schedules are identical. As N increases with γ held constant, ΔC increases as the dynamic schedule begins to outperform the static schedule. ΔC increases at a decreasing rate (i.e., the curves become closer together as N increases). The maximal value of ΔC is 10.6%. Even if N were to increase further beyond 15, it does not appear that the expected percentage cost saving would exceed 15%.

ΔC is at a minimum for the extreme values of γ (i.e., $\gamma = 0.1$ and $\gamma = 0.9$). An extreme value of γ indicates that the either the customer waiting cost or the server availability cost is significantly priorities (i.e., $c_W \gg c_S$ or $c_S \gg c_W$). If

one of the costs is heavily prioritised, there is little difference between the static and dynamic schedules, thus ΔC is small.

For each value of $N \geq 3$, the peak value of ΔC occurs at a middle value of γ. As N increases, the peak occurs at a larger value of γ. For $N = 3$ the peak occurs at $\gamma = 0.3$, whereas for $N = 15$ the peak occurs at $\gamma = 0.7$.

The difference between the dynamic and static schedule is most significant for middle values of γ (e.g., $\gamma \in [0.4, 0.7]$). For $\gamma = 0.1$, it doesn't appear that the expected percentage cost saving would exceed 5% indicating very little difference between the schedules.

Chapter 6

Simulation Studies

The previous chapter compared the static and dynamic schedules in terms of the expected cost of each schedule. We found for 15 customers, the dynamic schedule is about $5 - 10\%$ better than the static schedule depending on the value of γ. However, this does not provide any information about the other properties of each schedule. For example, one of the schedules could have a greater variability in customer waiting times.

To understand each schedule more fully, we use a simulation study. We simulated a million runs and measured the performance of both schedules on each run. Each run involved simulating the service times of the 15 customers assuming $\mu = 1$. All the simulations were completed assuming $\gamma = 0.5$.

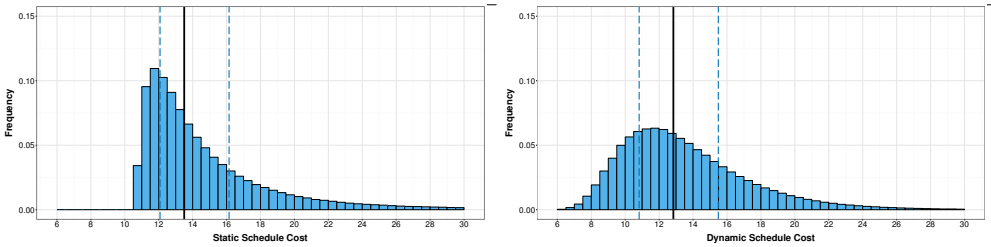
6.1 Schedule Cost

The mean costs of the static and dynamic schedule were 15.05 and 13.55 respectively. As expected, these mean costs closely match the expected costs of the schedules. Figure 6.1 plots the histograms of the costs of the static and dynamic schedules for each run of the simulation.

The distribution of the cost of the static schedule is right skewed. The peak of the distribution occurs at around 12, but the median is 13.5. In contrast, the distribution of the dynamic schedule is more symmetric with the peak occurring closer to the median of 12.8. The dynamic schedule is more flexible and thus less prone to runs with extremely high cost.

Figure 6.2 plots the histogram of the difference in costs between the two schedules. This is a plot of the cost of the static schedule minus the cost of the dynamic schedule for the same run.

While the expected cost of the static schedule is greater than the expected cost



(a) (b)

Figure 6.1: Histogram plot of the costs of the static (a) and dynamic (b) schedules for each simulation run where $\mu = 1$ and $\gamma = \frac{c_S}{c_S+c_W} = 0.5$. The black vertical line indicates the median and the blue dashes lines indicate the upper and lower quartiles.

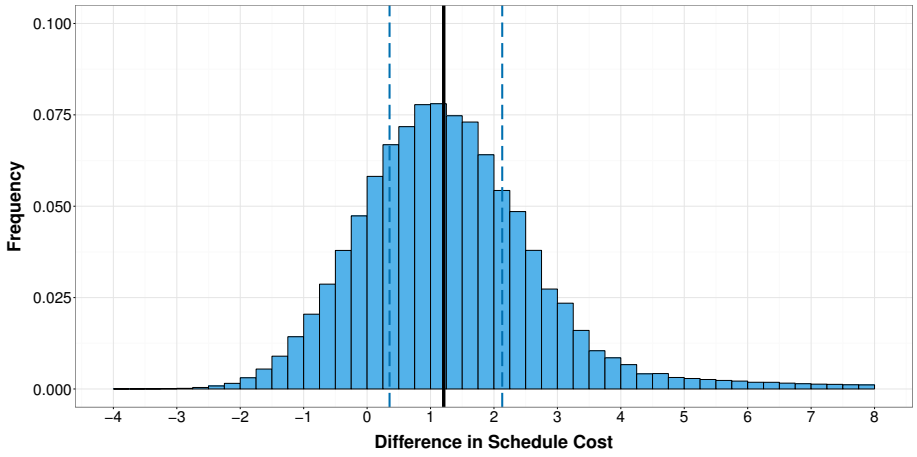


Figure 6.2: Histogram plot of the cost of the static schedule minus the cost of the dynamic schedule for each simulation run where $\mu = 1$ and $\gamma = \frac{c_S}{c_S+c_W} = 0.5$. The black vertical line indicates the median and the blue dashes lines indicate the upper and lower quartiles.

of the dynamic schedule, the static schedule outperforms the dynamic schedule for a proportion of the simulation runs. This is indicated by a negative cost difference in Figure 6.2. The 25-th percentile of the cost difference is 0.35. For more than 75% of runs, the static schedule has a larger cost than the dynamic schedule. The 75-th percentile is 2.13 (i.e., more than twice the mean service time). For a large proportion of the runs, the dynamic schedule significantly outperforms the static schedule.

6.2 Customer Arrival Times

We have observed that the dynamic schedule performs better than the static schedule both in expectation and for the majority of the simulation runs. The

next step is to attempt to understand how the dynamic schedule is able to outperform the static schedule. Figure 6.3 plots the average arrival time of the 15 customers for the two schedules.

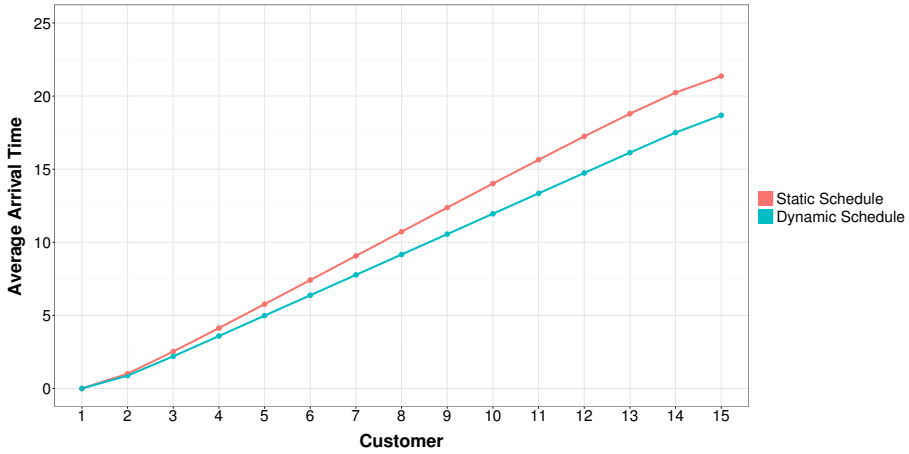


Figure 6.3: Plot of the average arrival time of the 15 customers for the static and dynamic schedules where $\mu = 1$ and $\gamma = \frac{c_S}{c_S + c_W} = 0.5$.

The static schedule has fixed arrival times, whereas the arrival times are chosen progressively for the dynamic schedule. For all 15 customers, their mean arrival time in the dynamic schedule is never later than their arrival time in the static schedule. The mean arrival times appear to be very similar for the first four or five customers. However, the later customers in the dynamic schedule appear to arrive significantly earler than the corresponding customers in the static schedule.

6.3 Customer Waiting Times

We would expect that the earlier arrival of the customers in the dynamic schedule would lead to those customers waiting longer. This hypothesis is examined by plotting the histogram of the difference in average customer waiting time between the two schedules in Figure 6.4. As before, this is a plot of the average waiting time of the static schedule minus the average waiting time of the dynamic schedule.

The 75-th percentile of the difference in average waiting time is 0.00. As expected, for the majority of the runs, the customers in the static schedule have a shorter average waiting time than the customers in the dynamic schedule.

However, the minimum difference in average waiting time is only -0.33, whereas the maximum is 8.57. It appears that in the simulation runs where

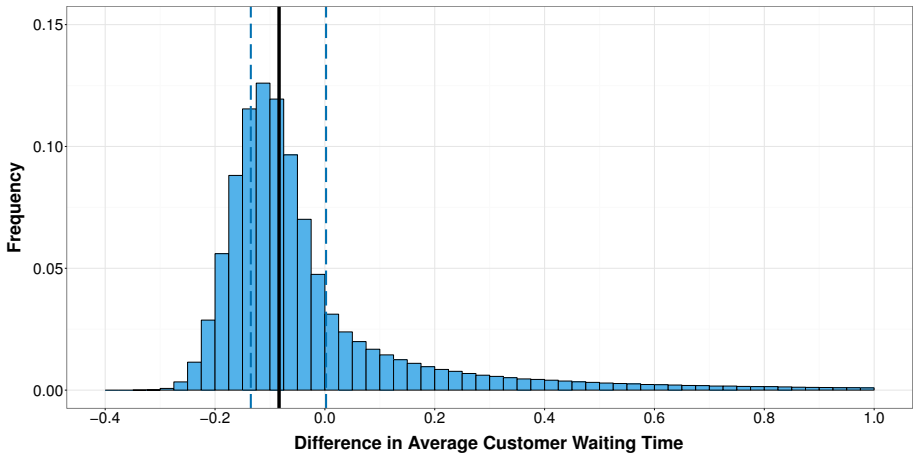


Figure 6.4: Histogram plot of the average waiting time for the static schedule minus the average waiting time for the dynamic schedule where $\mu = 1$ and $\gamma = \frac{c_S}{c_S+c_W} = 0.5$. The black vertical line indicates the median and the blue dashes lines indicate the upper and lower quartiles.

the dynamic schedule has a shorter waiting time than the static schedule, it tends to have a significantly shorter waiting time. The dynamic schedule is less prone to extremely long waiting times for the customers. This is expected behaviour as the dynamic schedule is able to react to customers having longer than expected service times and reschedule the later customers.

Another important consideration is the treatment of all the customers individually. Figure 6.5 plots the average waiting time of each customer individually.

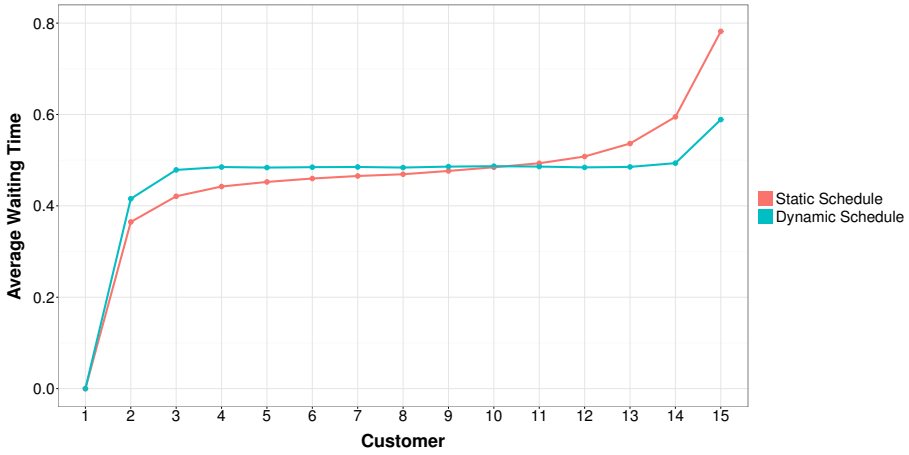


Figure 6.5: Plot of the average waiting time of the 15 customers for the static and dynamic schedules where $\mu = 1$ and $\gamma = \frac{c_S}{c_S+c_W} = 0.5$.

In both schedules, the average waiting time of the first customer is 0 as they are served immediately. The first few customers wait longer on average in the

dynamic schedule than the static schedule. In contrast, the later few customers tend to wait significantly longer in the static schedule.

For customer 3 to customer 14, their average waiting times are approximately constant in the dynamic schedule, but increasing in the static schedule. It appears that the dynamic schedule is significantly 'fairer' and able to spread the waiting time more evenly amongst the customers. The static schedule tends to favour the earlier customers.

Intuitively, customers prefer schedules where they are served immediately and do not have to wait. Figure 6.6 plots the proportion of runs where each customer does not have to wait.

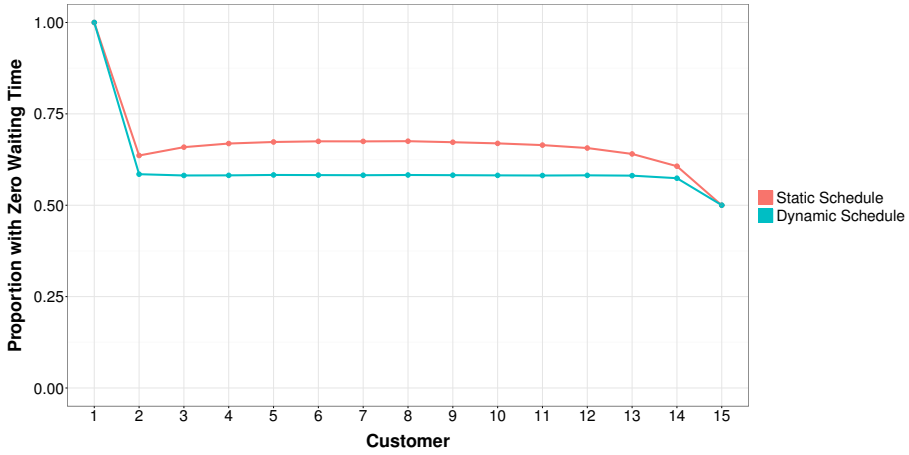


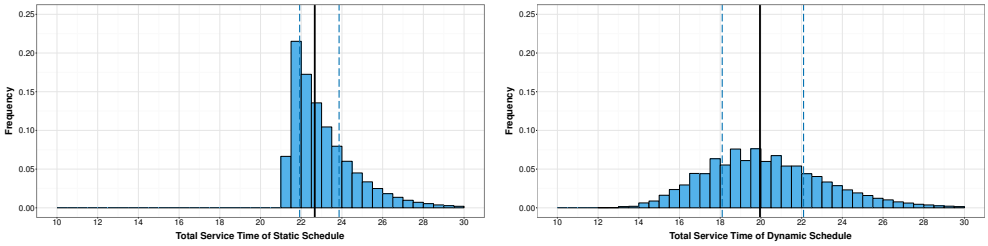
Figure 6.6: Plot of the proportion of all runs where the customer does not wait for each of the 15 customers for the static and dynamic schedules where $\mu = 1$ and $\gamma = \frac{c_S}{c_S + c_W} = 0.5$.

In both schedules, the first customer never waits and is served immediately in 100% of the runs. The dynamic schedule again appears to be fairer. The proportion of runs with no waiting time appears to be constant for all except the first and last customers. All customers (except the first and last) are served immediately a greater proportion of the time in the static schedule than the dynamic schedule. While this is advantageous for the customers, it leads to a longer idle time of the server and thus a longer server availability time.

6.4 Server Availability Time

The schedules do not only attempt to minimise customer waiting time, but they also attempt to minimise the server availability time. Figure 6.7 plots the histograms of the server availability times of the static and dynamic schedules for

each run of the simulation.



(a) (b)

Figure 6.7: Histogram plot of the server availability times for the static (a) and dynamic (b) schedules for each simulation run where $\mu = 1$ and $\gamma = \frac{c_S}{c_S+c_W} = 0.5$. The black vertical line indicates the median and the blue dashes lines indicate the upper and lower quartiles.

The median server availability time is 22.66 for the static schedule and 19.95 for the dynamic schedule. While the customer waiting times appear to be generally similar for both schedules, the dynamic schedule appears to attain its lower expected cost due to its lower server availability time. This agrees with the idea from Chapter 5 that the expected percentage cost saving is lowest where $\gamma = \frac{c_S}{c_S+c_W}$ is small. If the per unit time cost of the server availability time is small then the dynamic schedule's ability to reduce the server availability time is less effective at reducing the schedule cost.

Moreover, the minimum server availability time is significantly lower in the case of the dynamic schedule. The minimum server availability time is 21.37 for the static schedule and 12.11 for the dynamic schedule. This is especially surprising as both minimums occur in the same run of the simulation. The static schedule is significantly limited by the fixed arrival of the last customer at 21.36 time units. The dynamic schedule is able to adjust the customer arrival times and attain a broader range of server availability times that are lower on average.

Chapter 7

Conclusion

The goal of this thesis was to derive methods for determining the optimal customer arrival times in a single-server queue with punctual customers. In Chapter 3, we derived a similar method to Pegden and Rosenshine (1990) for determining the optimal static schedule. In Chapter 4, we derived an iterative procedure for determining the optimal dynamic schedule. This procedure determines the optimal arrival time of the next customer given the number of customers still to be scheduled and the number of customers currently in the system. In both schedules, the optimal schedule minimises a linear combination of the expected total customer waiting time and the expected server availability time.

In Chapter 3, we were able to derive algebraic expressions for the optimal policy for schedules with less than three customers. As expected, the interarrival time between the first two customers decreases as the server availability cost increases. In addition, we found numeric solutions for the case of scheduling 15 customers. We observed the dome-shape, which is a commonly observed property of static schedules. The optimal interarrival times increase for the initial few customers, remain constant for the majority of customers, and decrease for the final few customers.

In Chapter 4, we derived exact algebraic expressions for the optimal dynamic schedule where there are less than three customers. These dynamic schedules exact match the optimal static schedules found in Chapter 3. For more than two customers, the optimal dynamic schedule consists of the optimal policies at each of the possible states during service.

We considered the case of scheduling 15 customers using a dynamic schedule. We found that the optimal interarrival time of the next customer appears to only depend on the number currently in the system provided that it is not the final customer being scheduled. If this hypothesis was proved in the general case, then

it would have several potential benefits. First, it would be significantly easier to apply such a schedule in practice as the scheduler is only required to know a few optimal policies. Moreover, it would be more efficient to compute the optimal dynamic schedule as it would involve solving for a smaller number of unknown variables. However, more work is required to confirm this hypothesis.

In Chapter 5, we compared the expected cost of the two schedules. As expected, the static schedule never has a smaller expected cost than the dynamic schedule. However, we found that the expected cost difference only appears to be noticeable when scheduling more than five customers. The expected percentage cost saving increases as the number of customers to be scheduled increases, but at a decreasing rate. For scheduling 15 customers with $\gamma = 0.5$, the expected percentage cost saving is generally between 5 and 10%. From our results, it does not appear that the expected percentage cost saving would ever exceed 15%.

In Chapter 6, we applied simulation to get a fuller understanding of the differences between the two schedules. The dynamic schedule is able to adapt during service, and is thus less prone to services with extremely high cost in comparison to the static schedule. The dynamic schedule generally schedules customers to arrive earlier than the static schedule. Customers thus often wait late longer in the dynamic schedule than the static schedule. However, the dynamic schedule appears to be fairer as the average waiting time is approximately constant for the majority of the customers. The dynamic schedule appears to attain its lower expected cost due to its considerably lower server availability time. For 15 customers where $\gamma = 0.5$, the median server availability time is 12% lower for the dynamic schedule.

The results presented in this thesis suggest that the dynamic schedule can often significantly outperform the static schedule. Models that ignore the potential ability to reschedule scheduled appear to be too restrictive. However, this work should be extended to more clearly understand how the rescheduling ability should be included.

A highly beneficial extension of the dynamic schedule presented in Chapter 4 would include the idea of a minimum notice period. In many situations, customers are not able to arrive immediately. The dynamic schedule would be significantly more applicable if it could include a minimum value on the scheduled interarrival times. For the minimum notice period to be properly included in the schedule, the schedule must have the ability to schedule multiple customers at once rather than needing to schedule the customers one-by-one.

Other extensions on the schedules prevented in this thesis would involve ad-

justments to the objective function. Many papers include the server's idle time in the objective. Both the objective function for the static schedule in Chapter 3 and the expected transition cost in Chapter 4 can be easily adjusted to include the cost of the server's idle time.

Another extension on the objective function could include the cost of overtime. The service could have a desired finish time, and there is a (high) cost if the service exceeds this time. This extension would be beneficial to real world services. However, applying it requires knowledge of this desired finish time parameter, which is likely heavily dependent on the specific service under consideration.

Appendix A

Dynamic Schedule Derivation

A.1 Transition Probability

This is a derivation of the transition probability $p_a(k, j)$. This is the probability that there are $k - (j - 1)$ departures from a queue over a time units assuming the queue initially has k customers and no new arrivals. The customer service times are iid exponential random variables with mean μ.

The probabililty is most easily derived on a case by case basis.

A.1.1 Case 1 $k = 0$

The first case is that the queue initially has no customers. For any $a \geq 0$, there will be no departures from the queue over a time units.

$$p_a(k, j) = \mathbb{1}(j = 1) \tag{A.1}$$

A.1.2 Case 2 $k \geq 1, j = 1$

Denote the service times of the k customers as $S_1, \ldots, S_k$. If $j = 1$, then $p_a(k, j)$ is the probability of k departures from the queue over a time units. The sum $\sum_{i=1}^{k} S_i$ has an Erlang distribution with cdf $F(a; k)$.

$$p_a(k, j) = \mathbb{P}\left(\sum_{i=1}^{k} S_i \leq a\right) = F(a; k) \tag{A.2}$$

A.1.3 Case 3 $k \geq 2, 2 \leq j \leq k$

For $2 \leq j \leq k$, $p_a(k, j)$ is the probability that the total service time of the first $k - (j - 1)$ customers is less that a, and the total service time of the first

$k-(j-1)+1$ customers is greater than a.

$$\begin{aligned} p_a(k,j) &= \mathbb{P}\left(\sum_{i=1}^{k-(j-1)} S_i \leq a, \sum_{i=1}^{k-(j-1)+1} S_i > a\right) \\ &= \mathbb{P}\left(\sum_{i=1}^{k-(j-1)} S_i \leq a, S_{k-(j-1)+1} > a - \sum_{i=1}^{k-(j-1)} S_i\right) \end{aligned} \tag{A.3}$$

Condition the probability on $\sum_{i=1}^{k-(j-1)} S_i = z$, which has pdf $f(z; k-(j-1))$ and integrate over all possible values of z.

$$\begin{aligned} p_a(k,j) &= \int_0^\infty \mathbb{P}(z \leq a, S_{k-(j-1)+1} > a - z) f\big(z; k-(j-1)\big)\, dz \\ &= \int_0^a \mathbb{P}(S_{k-(j-1)+1} > a - z) f\big(z; k-(j-1)\big)\, dz \\ &= \frac{1}{(k-j)!}\left(\frac{1}{\mu}\right)^{k-j+1} \mathrm{e}^{\frac{-a}{\mu}} \int_0^a z^{k-j}\, dz \\ &= \frac{1}{(k-j+1)!}\left(\frac{a}{\mu}\right)^{k-j+1} \mathrm{e}^{\frac{-a}{\mu}} \\ &= F(a; k-j+1) - F(a; k-j+2) \end{aligned} \tag{A.4}$$

A.1.4 Case 4 $k \geq 1, j = k+1$

For $j = k+1$, $p_a(k,j)$ is the probability that the service time of the first customer is longer than a time units.

$$p_a(k,j) = \mathbb{P}(S_1 > a) = 1 - \mathbb{P}(S_1 \leq a) = 1 - F(a;1) \tag{A.5}$$

A.1.5 All Other Cases

All other cases have zero probability.

$$p_a(k,j) = 0 \tag{A.6}$$

A.1.6 Summary

These results can be summarised as:

$$p_a(k,j) = \begin{cases} \mathbb{1}(j=1) & \text{for } k=0 \\ F(a;k) & \text{for } k \geq 1, j=1 \\ F(a;k-j+1) - F(a;k-j+2) & \text{for } k \geq 2, 2 \leq j \leq k \\ 1 - F(a;1) & \text{for } k \geq 1, j=(k+1) \\ 0 & \text{otherwise} \end{cases} \tag{A.7}$$

A.2 Expected Transition Cost

This is a derivation of the expected transition cost $R_a(k,j)$. This is the expected cost of transitioning from the state (n,k) to the state $(n-1,j)$ over a time units if the next customer is scheduled to arrive in a time units. The expected cost is a linear combination of the expected total customers' waiting times and the expected server availability time. The per unit time costs c_W and c_S are defined as in Chapter 3.

In a similar way to the transition probability, the cost is most easily derived on a case by case basis.

A.2.1 Case 1 $k \in \{0,1\}$

The first case is that the system initially has either no customers or only a single customer. In this case, no customers are waiting during the transition, so the total customers waiting time is zero. The only cost is the cost of expected server availability time during the transition. The server is available for the entire transition, so the server availability time is a.

$$R_a(k,j) = c_S a \tag{A.8}$$

A.2.2 Case 2 $k \geq 2, j = 1$

If $j=1$, then all k customers finish service during the transition. This implies that $\sum_{n=1}^{k} S_n \leq a$. The total customers' waiting time is a linear combination of the service times given that the summation of the service time is smaller than a. The server is still available for the entire transition, so the server availability

time is a.

$$\begin{aligned} R_a(k,j) &= c_W \sum_{i=2}^{k} \mathbb{E}\left[\sum_{l=1}^{i-1} S_l \;\middle|\; \sum_{n=1}^{k} S_n \le a\right] + c_S a \\ &= c_W \mathbb{E}\left[S_1 \;\middle|\; \sum_{n=1}^{k} S_n \le a\right] \sum_{i=2}^{k} (i-1) + c_S a \\ &= c_W \frac{k(k-1)}{2} \mathbb{E}\left[S_1 \;\middle|\; \sum_{n=1}^{k} S_n \le a\right] + c_S a \\ &= c_W \frac{(k-1)}{2} \mathbb{E}\left[\sum_{n=1}^{k} S_n \;\middle|\; \sum_{n=1}^{k} S_n \le a\right] + c_S a \end{aligned} \tag{A.9}$$

The term $\mathbb{E}\left[\sum_{n=1}^{k} S_n \;\middle|\; \sum_{n=1}^{k} S_n \le a\right]$ is one of the conditional expectations defined in Chapter 4.

$$R_a(k,j) = c_W \frac{G(a;k)(k-1)}{2} + c_S a \tag{A.10}$$

A.2.3 Case 3 $k \ge 2, 2 \le j \le k$

For $2 \le j \le k$, the first $k-(j-1)$ customers finish service during the transition, but customer $k-(j-1)+1$ does not. This implies that $\sum_{n=1}^{k-(j-1)} S_n \le a$ and $\sum_{n=1}^{k-(j-1)+1} S_n > a$. The last $j-2$ customers wait for the entire transition. In addition, the server is available for the entire transition.

$$\begin{aligned} R_a(k,j) = c_W &\sum_{i=2}^{k-(j-2)} \mathbb{E}\left[\sum_{l=1}^{i-1} S_l \;\middle|\; \sum_{n=1}^{k-(j-1)} S_n \le a, \sum_{n=1}^{k-(j-1)+1} S_n > a\right] \\ &+ c_W \sum_{i=1}^{j-2} a + c_S a \\ = c_W &\mathbb{E}\left[S_1 \;\middle|\; \sum_{n=1}^{k-(j-1)} S_n \le a, \sum_{n=1}^{k-(j-1)+1} S_n > a\right] \sum_{i=2}^{k-(j-2)} (i-1) \\ &+ c_W a(j-2) + c_S a \end{aligned}$$

$$
\begin{aligned}
R_a(k,j) &= c_W \frac{\big[k-(j-1)\big]\big[k-(j-2)\big]}{2} \mathbb{E}\left[S_1 \,\middle|\, \sum_{n=1}^{k-(j-1)} S_n \le a, \sum_{n=1}^{k-(j-1)+1} S_n > a \right] \\
&\quad + c_W a(j-2) + c_S a \\
&= c_W \frac{\big[k-(j-2)\big]}{2} \mathbb{E}\left[\sum_{n=1}^{k-(j-1)} S_n \,\middle|\, \sum_{n=1}^{k-(j-1)} S_n \le a, \sum_{n=1}^{k-(j-1)+1} S_n > a \right] \\
&\quad + c_W a(j-2) + c_S a
\end{aligned} \tag{A.11}
$$

The term $\mathbb{E}\left[\sum_{n=1}^{k-(j-1)} S_n \,\middle|\, \sum_{n=1}^{k-(j-1)} S_n \le a, \sum_{n=1}^{k-(j-1)+1} S_n > a \right]$ is another of the conditional expectations defined in Chapter 4.

$$
\begin{aligned}
R_a(k,j) &= c_W \left[\frac{H\big(a; k-(j-1)\big)\big[k-(j-2)\big]}{2} + a(j-2) \right] + c_S a \\
&= c_W \left[\frac{a\big[k-(j-1)\big]}{2} + a(j-2) \right] + c_S a \\
&= c_W \frac{a(k+j-3)}{2} + c_S a
\end{aligned} \tag{A.12}
$$

A.2.4 Case 4 $k \ge 2, j = (k+1)$

For $j = k+1$, no customers finish service during the transition. All customers except the first customer wait for the entire transition. The server is available for the entire transition.

$$
R_a(k,j) = c_W \sum_{i=1}^{k-1} a + c_S a = c_W a(k-1) + c_S a \tag{A.13}
$$

A.2.5 Summary

In order to compare the dynamic schedule with the static schedule, need to scale these costs by dividing by $(c_S + c_W)$ and defining $\gamma = \frac{c_S}{c_S + c_W}$. These results can

be summarised as:

$$R_a(k,j) = \begin{cases} \gamma a & \text{for } k \in \{0,1\} \\ (1-\gamma)\frac{G(a;k)(k-1)}{2} + \gamma a & \text{for } k \geq 2, j = 1 \\ (1-\gamma)\frac{a(k+j-3)}{2} + \gamma a & \text{for } k \geq 2, 2 \leq j \leq k+1 \end{cases} \tag{A.14}$$

Appendix B

Python Code

B.1 Static Schedules

```
import numpy as np
import math
import scipy.optimize as optimize

import os
from multiprocessing import Pool

# distribution of number in system
def probSystem(i, j, x, mu):
    if (i == 1 and j == 0):
        return 1

    elif (i >= 2 and j == 0):
        value = 0
        for k in range(1, i):
            internal = 0
            for l in range(k):
                internal += (x[i-2]**l / (mu**l *
                            math.factorial(l))) * np.exp(-x[i-2] /
                            mu)

            value += probSystem(i - 1, k - 1, x, mu) * (1 -
                            internal)

        return value
```

```
    elif (i >= 2 and j >= 1):
        value = 0
        for k in range(i - j):
            value += probSystem(i - 1, j + k - 1, x, mu) * \
                     (x[i-2]**k / (mu**k *
                     math.factorial(k))) * np.exp(-x[i-2] / mu)

        return value

# expected waiting time
def waitTime(i, x, mu):
    if (i == 1):
        return (0)

    elif (i >= 2):
        value = 0
        for j in range(1, i):
            value += (j * mu) * probSystem(i, j, x, mu)

        return value

# objective function
def phi(x, gamma, mu):
    # if any x is negative during optimisation, set to zero
    for i in range(len(x)):
        if (x[i] < 0):
            x[i] = 0

    n = len(x) + 1
    value = 0

    for i in range(1, n + 1):
        value += (1 - gamma) * waitTime(i, x, mu)

    for i in range(1, n):
        value += gamma * x[i-1]

    value += gamma * waitTime(n, x, mu) + gamma * mu

    return value
```

```
# function to find optimal solution
def solver(args):
    n, gamma, mu, fname = args

    # find bounds
    bnds = ()
    for i in range(n - 1):
        bnds += ((0, None),)

    # minimise
    res = optimize.minimize(fun = phi, x0 = [0]*(n - 1),
            args = (gamma, mu), method = "L-BFGS-B", bounds = bnds)

    # write output
    with open(fname, "a+") as outputFile:
        outputFile.write(str(n) + "," + str(gamma) + "," + str(mu) +
                "," + repr(res.x) + "," + str(res.fun) + "\n")

if __name__ == "__main__":
    mu = 1

    # output file name
    output = "Static_Output.txt"

    nVec = np.arange(2, 16, 1)
    gammaVec = np.arange(0.1, 1, 0.1)

    args = [(n, gamma, mu, output) for n in nVec
                for gamma in gammaVec]

    # run on 12 cores
    p = Pool(12)

    # delete output file if exists
    if os.path.exists(output):
        os.remove(output)

    # create header
    with open(output, "w+") as outputFile:
        outputFile.write("n,gamma,mu,x,cost\n")

    p.map(solver, args)
```

B.2 Dynamic Schedules

```
import numpy as np
from scipy.stats import poisson
import scipy.optimize as optimize

import os
from multiprocessing import Pool

# distribution functions
def cdf(a, r, mu):
    if (a > 0):
        return poisson.sf(r - 1, a / mu)

    elif (a == 0):
        return 0

# transition probability
def transProb(a, k, j, mu):
    if (k == 0):
        return int(j == 1)

    elif (k >= 1 and j == 1):
        return cdf(a, k, mu)

    elif (k >= 2 and 2 <= j and j <= k):
        return cdf(a, k - j + 1, mu) - cdf(a, k - j + 2, mu)

    elif (k >= 1 and j == (k + 1)):
        return 1 - cdf(a, 1, mu)

    else:
        return 0

# conditional expectation
def condExp(a, r, mu):
    if (a > 0):
        return mu * r * cdf(a, r + 1, mu) / cdf(a, r, mu)

    elif (a == 0):
        return 0
```

```
44 # expected transition cost
45 def transCost(a, k, j, gamma, mu):
46     if (k == 0 or k == 1):
47         return gamma * a
48 
49     elif (k >= 2 and j == 1):
50         return (1 - gamma) * condExp(a, k, mu) * (k - 1) / 2 + \
51                     gamma * a
52 
53     elif (k >= 2 and 2 <= j and j <= k + 1):
54         return (1 - gamma) * a * (k + j - 3) / 2 + gamma * a
55 
56 # schedule cost
57 def cost(a, n, k, gamma, mu, computedDict):
58     # if a is negative during optimisation, set to zero
59     if (a < 0):
60         a = 0
61 
62     internal = 0
63     for j in range(1, k + 2):
64         internal += transProb(a, k, j, mu) * \
65                         (transCost(a, k, j, gamma, mu) +
66                         optimalCost(n - 1, j, gamma, mu,
67                         computedDict)[0])
68 
69     return internal
70 
71 # optimal schedule cost
72 def optimalCost(n, k, gamma, mu, computedDict):
73     if (n, k, gamma, mu) in computedDict:
74         return (computedDict[(n, k, gamma, mu)][0],
75                 computedDict[(n, k, gamma, mu)][1], computedDict)
76 
77     if (n == 0):
78         newCost = (1 - gamma) * mu * k * (k - 1) / 2 + \
79                         gamma * k * mu
80         time = float('nan')
81 
82         computedDict[(n, k, gamma, mu)] = [newCost, time]
83 
84         return newCost, time, computedDict
85 
86 
87 
88 
```

```
    elif (n >= 1 and k == 0):
        newCost = optimalCost(n - 1, 1, gamma, mu, computedDict)[0]
        time = 0

        computedDict[(n, k, gamma, mu)] = [newCost, time]

        return newCost, time, computedDict

    elif (n >= 1 and k >= 1):
        res = optimize.minimize(fun = cost, x0 = [0],
                    args = (n, k, gamma, mu, computedDict),
                    method = "L-BFGS-B", bounds = ((0, None),))

        newCost = res.fun[0]
        time = res.x[0]

        computedDict[(n, k, gamma, mu)] = [newCost, time]

        return newCost, time, computedDict

# function to find optimal solution
def solver(args):
    N, gamma, mu, fname = args

    allResults = dict()

    for n in range(N + 1):
        for k in range(N - n + 1):
            singleCost, time, allResults = optimalCost(n, k, gamma,
                                                mu, allResults)

            # write output
            with open(fname, "a+") as outputFile:
                outputFile.write(str(n) + "," + str(k) + "," +
                    str(gamma) + "," + str(mu) + "," + str(time) +
                    "," + str(singleCost) + "\n")

if __name__ == "__main__":
    mu = 1
    N = 15

    # output file name
    output = "Dynamic_Output.csv"

    gammaVec = np.arange(0.1, 1, 0.1)
```

```
134     args = [(N, gamma, mu, output) for gamma in gammaVec]
135 
136     # run on two cores
137     p = Pool(2)
138 
139     # delete output file if exists
140     if os.path.exists(output):
141         os.remove(output)
142 
143     # create header
144     with open(output, "w+") as outputFile:
145         outputFile.write("n,k,gamma,mu,a,cost\n")
146 
147     p.map(solver, args)
```

B.3 Simulation

```
import os
import csv
import numpy as np

# set seed
seed = 1984
np.random.seed(seed)

mu = 1
N = 15
gamma = 0.5

numRuns = 10**6

outputName = "Simulation_Output.csv"

# store arrival times, service start and end times
# for each customer, run and schedule
arrivalTime = np.zeros(shape = (2,numRuns,N))
serviceStart = np.zeros(shape = (2,numRuns,N))
serviceEnd = np.zeros(shape = (2,numRuns,N))

# store waiting time for each customer, run and schedule
waitingTime = np.zeros(shape = (2,numRuns,N))

# store total waiting time, total service time, total idle time and
# total cost for each run and schedule
totalWaitTime = np.zeros(shape = (2,numRuns))
totalServiceTime = np.zeros(shape = (2,numRuns))
totalIdleTime = np.zeros(shape = (2,numRuns))
totalCost = np.zeros(shape = (2,numRuns))

# expected cost of each schedule
cost = np.zeros(shape = 2)

# read in static interarrival times
with open("Static_Output.csv", 'r') as outputFile:
    reader = csv.reader(outputFile)
    for row in reader:
        if (row[0:3] == [str(N),str(gamma),str(mu)]):
            interarrivalStatic = [float(val) for val in row[3:N+2]]
            cost[0] = float(row[-1])
            break
```

```
# read in dynamic interarrival times
interarrivalDynamic = dict()
with open("Dynamic_Output.csv", 'r') as outputFile:
    reader = csv.reader(outputFile)
    for row in reader:
        if (row[2] == str(gamma) and row[3] == str(mu)):
            if (int(row[0]) == N and int(row[1]) == 0):
                interarrivalDynamic[(int(row[0]), int(row[1]))] = \
                                    float(row[4])
                cost[1] = float(row[5])

            elif (int(row[0]) + int(row[1]) <= N):
                interarrivalDynamic[(int(row[0]), int(row[1]))] = \
                                    float(row[4])

# generate service time of all N customers (for all runs)
serviceTime = np.random.exponential(scale = mu, size = (numRuns,N))

for run in range(numRuns):
    # calculate arrival time of all N customers for static schedule
    arrivalTime[0,run] = np.append(0, np.cumsum(interarrivalStatic))

    # arrival time for first customer in dynamic schedule
    arrivalTime[1,run,0] = interarrivalDynamic[(N, 0)]

    # calculate time service starts and time service ends
    # for first customer
    for sch in range(2):
        serviceStart[sch,run,0] = arrivalTime[sch,run,0]
        serviceEnd[sch,run,0] = serviceStart[sch,run,0] + \
                                        serviceTime[run,0]

    for i in range(1, N):
        # arrival time for customer i in dynamic schedule
        arrivalTime[1,run,i] = arrivalTime[1,run,i-1] + \
            interarrivalDynamic[(N-i,
            sum(serviceEnd[1,run,0:i] > arrivalTime[1,run,i-1]))]

        # calculate time service starts and time service ends
        for sch in range(2):
            serviceStart[sch,run,i] = max(arrivalTime[sch,run,i],
                                          serviceEnd[sch,run,i-1])
            serviceEnd[sch,run,i] = serviceStart[sch,run,i] + \
                                          serviceTime[run,i]

```

```
    # calculate waiting time of each customer
    for i in range(N):
        for sch in range(2):
            waitingTime[sch,run,i] = serviceStart[sch,run,i] - \
                                     arrivalTime[sch,run,i]

    # calculate total waiting time and total service time
    for sch in range(2):
        totalWaitTime[sch,run] = sum(waitingTime[sch,run])
        totalServiceTime[sch,run] = serviceEnd[sch,run,N-1]

    # calculate total idle time
    for sch in range(2):
        totalIdleTime[sch,run] = totalServiceTime[sch,run] - \
                                 sum(serviceTime[run])

    # calculate total cost
    for sch in range(2):
        totalCost[sch,run] = gamma * totalServiceTime[sch,run] + \
                             (1 - gamma) * totalWaitTime[sch,run]

print('For Static Schedule:')
print('Expected Cost is ' + str(cost[0]))
print('Mean Cost is ' + str(np.mean(totalCost[0])))

print('')

print('For Dynamic Schedule:')
print('Expected Cost is ' + str(cost[1]))
print('Mean Cost is ' + str(np.mean(totalCost[1])))

# delete output file if exists
if os.path.exists(outputName):
    os.remove(outputName)

# create header
header = ['schedule','run']

for i in range(1, N + 1):
    header += ['AT[' + str(i) + ']']

for i in range(1, N + 1):
    header += ['WT[' + str(i) + ']']

header += ['TWT', 'TST', 'TIT', 'Cost']
```

```
134 with open(outputName, "w+") as outputFile:
135     writer = csv.writer(outputFile)
136     writer.writerow(header)
137 
138 for sch in range(2):
139     for run in range(numRuns):
140         if (sch == 0):
141             row = ['static']
142         else:
143             row = ['dynamic']
144 
145         row += [run]
146 
147         row += [arrivalTime[sch,run,i] for i in range(N)]
148         row += [waitingTime[sch,run,i] for i in range(N)]
149 
150         row += [totalWaitTime[sch,run], totalServiceTime[sch,run],
151                 totalIdleTime[sch,run], totalCost[sch,run]]
152 
153         # write output
154         with open(outputName, "a+") as outputFile:
155             writer = csv.writer(outputFile)
156             writer.writerow(row)
```

Bibliography

Babes, Malika and GV Sarma (1991). "Out-patient queues at the Ibn-Rochd health centre". *Journal of the Operational Research Society* 42.10, pp. 845–855.

Bailey, Norman TJ (1952). "A study of queues and appointment systems in hospital out-patient departments, with special reference to waiting-times". *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 185–199.

Bellman, Richard (1957). *A Markovian decision process*. Tech. rep. DTIC Document.

Bennett, Joanne C and DJ Worthington (1998). "An example of a good but partially successful OR engagement: Improving outpatient clinic operations". *Interfaces* 28.5, pp. 56–69.

Byrd, Richard H et al. (1995). "A limited memory algorithm for bound constrained optimization". *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.

Cayirli, Tugba and Emre Veral (2003). "Outpatient scheduling in health care: A review of literature". *Production and Operations Management* 12.4, pp. 519–549.

Chow, Timothy Y (1999). "What is a closed-form number?" *The American Mathematical Monthly* 106.5, pp. 440–448.

DeLaurentis, Po-Ching et al. (2006). "Open access appointment scheduling - An experience at a community clinic". *IIE Annual Conference*. Institute of Industrial Engineers.

Erdogan, S Ayca and Brian Denton (2011). "Dynamic appointment scheduling of a stochastic server with uncertain demand". *INFORMS Journal on Computing* 25.1, pp. 116–132.

Fiems, Dieter, Ger Koole, and Philippe Nain (2007). "Waiting times of scheduled patients in the presence of emergency requests". *Technisch Rapport*.

Fomundam, Samuel and Jeffrey W Herrmann (2007). *A survey of queuing theory applications in healthcare*. University of Maryland, The Insitute for Systems Research.

Goldsmith, Jeff (1989). "A radical prescription for hospitals". *Harvard Business Review*.

Green, Linda (2006). "Queueing analysis in healthcare". *Patient flow: reducing delay in healthcare delivery*. Springer, pp. 281–307.

Gupta, Ishwar, Juan Zoreda, and Nathan Kramer (1971). "Hospital manpower planning by use of queueing theory". *Health Services Research* 6.1, pp. 76–82.

Ho, Chrwan-Jyh and Hon-Shiang Lau (1992). "Minimizing total cost in scheduling outpatient appointments". *Management Science* 38.12, pp. 1750–1764.

Huarng, Fenghueih and Mong Hou Lee (1996). "Using simulation in out-patient queues: A case study". *International Journal of Health Care Quality Assurance* 9.6, pp. 21–25.

Kao, Edward PC and Grace G Tung (1981). "Bed allocation in a public health care delivery system". *Management Science* 27.5, pp. 507–520.

Mendel, Sharon (2006). "Scheduling arrivals to queues: A model with no-shows". MA thesis. Tel-Aviv University.

Mondschein, Susana V and Gabriel Y Weintraub (2003). "Appointment policies in service operations: A critical analysis of the economic framework". *Production and Operations Management* 12.2, pp. 266–286.

O'Keefe, Robert M (1985). "Investigating outpatient departments: Implementable policies and qualitative approaches". *Journal of the Operational Research Society* 36.8, pp. 705–712.

Pegden, Claude Dennis and Matthew Rosenshine (1990). "Scheduling arrivals to queues". *Computers & Operations Research* 17.4, pp. 343–348.

Rockart, John F and Paul B Hofmann (1969). "Physician and patient behavior under different scheduling systems in a hospital outpatient department". *Medical Care* 7.6, pp. 463–470.

Sniedovich, M (1986). "A new look at Bellman's Principle of Optimality". *Journal of Optimization Theory and Applications* 49.1, pp. 161–176.

Stein, William E and Murray J Côté (1994). "Scheduling arrivals to a queue". *Computers & Operations Research* 21.6, pp. 607–614.

Walter, SD (1973). "A comparison of appointment schedules in a hospital radiology department". *British Journal of Preventive & Social Medicine* 27.3, pp. 160–167.

Wang, P Patrick (1993). "Static and dynamic scheduling of customer arrivals to a single-server system". *Naval Research Logistics (NRL)* 40.3, pp. 345–360.

Wang, P Patrick (1997). "Optimally scheduling N customer arrival times for a single-server system". *Computers & Operations Research* 24.8, pp. 703–716.