Task Allocation Policies for State Dependent Queues

by

Christine Chiu Hsia Siew

Bachelor of Science in Engineering in Aerospace Engineering
University of Michigan (2008)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Author ....
Department of Aeronautics and Astronautics
Jan 15, 2011

Certified by..................... ....
Prof. Emilio Frazzoli
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by.................................
Prof. Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Task Allocation Policies for State Dependent Queues

by

Christine Chiu Hsia Siew

Submitted to the Department of Aeronautics and Astronautics
on Jan 15, 2011, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Consider a model of a dynamical queue with deterministic arrival and service rates, where the service rate depends on the server utilization history. This proposed queueing model occurs in many practical situations, for example in human-in-the-loop systems where widely accepted empirical laws describe human performance as a function of mental arousal, which increases when the human is working on a task and decreases otherwise.

Formal methods for task management in state-dependent dynamical queues are gathering increasing attention to improve the efficiency of such systems. The focus of this research is hence to design maximally stabilizing task release control policies to maximize the useful throughput of such a system. Assuming that the error probability of a server is also related to its state, the useful throughput can be defined as the number of successfully completed tasks per unit time. Monitoring of both service and error rates is particularly typical in the realm of human-in-the-loop and production systems.

This research focuses on developing policies to minimize both these penalty measures. For a server with deterministic service rate, the optimal policy is found to be a threshold policy that releases a task to the server only when the server state is less than or equal to a certain threshold. Assuming homogeneous tasks that bring in the same deterministic amount of work to be done, it can be shown that an appropriate threshold policy is maximally stabilizing and that this threshold value can be uniquely determined. This work is then further extended to the case when the server behaves stochastically and verified using simulation. Finally, a proof-of-concept experiment is proposed and developed to test the feasibility of the proposed theoretical policies in real-world settings. The experiment consisted of completing multiple-choice verbal analogy questions and the results confirm the effect of workload control in improving human performance.

Thesis Supervisor: Prof. Emilio Frazzoli
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

Firstly, I would like to thank my advisor Professor Emilio Frazzoli for his guidance and patience with me throughout my research journey. His patience and willingness to help me through these very difficult 2 years have inspired and motivated me to improve myself. As a new graduate student with very little knowledge of the area I was diving into, Professor Frazzoli's guidance both in class and in the lab has enabled me to grow both academically and character-wise and I am very grateful for the help he has provided.

Secondly, I would also like to thank Dr. Ketan Savla who has helped me immensely in my research journey, particularly in teaching me how research is done. The many times I have brought my lofty ideas to him only to have him word them in a more achievable manner have helped me in becoming a better researcher and thinker. Indeed, it is through my many discussions with him that I am a better control theorist today than I was when I first came to MIT.

Next, I would like to thank all the members of the Aerospace Robotics and Embedded Systems Laboratory (ARES) for their help and friendship in 32-D712. Be it rain or shine, day or night, there was always a friendly face there to provide company in the lab. Special thanks goes out to Tom Temple for being my TA in my autonomy course at MIT and building my interest in this area and to Vu Anh Huynh for being a sounding board for all my ideas despite his busy courseload.

As this chapter of my life draws to a close, I would also like to thank all professors, lecturers and teaching assistants from MIT and the University of Michigan for teaching me everything I now know. Learning in both schools, the former for my Masters and the latter for my undergraduate, has been challenging but fulfilling and it would not have been possible without the strong academic support both schools have.

Finally, this thesis would not have been possible without support from my friends and family back home in Singapore. My mum, sister and aunt have been a great source of strength throughout my days abroad and I am thankful for their love and belief in me, even at times when I did not believe in myself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There is an increasing demand for better resource allocation in many sectors of today's society, particularly with the economical benefits associated with having higher efficiency on production floors. In fact, during the last two decades, workload-based task allocation policies have become a popular subject both in research and in practice. This is due to the fact that in many jobshop-like production situations, the main management issue is the control of work order throughput times in order to achieve a high level of customer service (such as short wait times and met deadlines), in combination with an optimal level of resource utilization [11, 16]. Moreover, scarcity of resources often give rise to congestion phenomena, commonly known as production bottlenecks. The reduction of congestion, optimization of resource allocation and development of workload-based task allocation policies hence form an emerging realm of queueing theory and performance analysis that needs to be addressed.

Studies of queueing systems typically assume a service rate that is uniform over time, e.g., as described by a stochastic variable with a fixed distribution. There are however various application areas where this assumption may not be valid, such as communication networks, production systems and even in Human Supervisory Control (HSC) settings. In such settings, the system time may be affected not only by the level of congestion, but also by how much work the server has processed in the past. Queueing models with such workload-dependent rates will thus form the main subject of this thesis and the aim of the rest of this thesis is to develop allocation policies to achieve optimal performance metrics while preventing backlog from accumulating in the queue.

1.2 Classical Queueing Models

The most basic queueing system consists of the following 6 basic characteristics: (1) an arrival process of customers, (2) a service pattern for the server(s), (3) a queue discipline, (4) a queue system capacity, (5) a number of service channels and (6) a number of service stages. A typical queueing system would hence consist of a customer arriving for service, waiting in the queue if service is not immediate, and leaving the system after having been served. The term 'customer' is however used in a general sense and indicates any arriving task that can be serviced by the server, for example a computer program that needs to be executed.

Queueing theory was developed at the beginning of the 20^{th} century to provide models to predict the behavior of systems that attempt to provide service for demands that arise randomly. The earliest problems studied were those of telephone traffic congestion and the first investigator was the Danish mathematician A.K. Erlang, who in 1909 published 'The Theory of Probabilities and Telephone Conversations'. Erlang introduced the notion of stability equilibrium and balance-of-state equations into the realm of queues as he sought to find the optimal number of telephone operators needed to handle a certain volume of telephone calls. These methods and his classic Erlang Loss formulae are still in use today and are the first known considerations of the optimization of a queueing system. Since then, queueing theory has grown into a field with many valuable applications in the areas of operations research, traffic flow, scheduling and facility design. For a summary of the evolution of queueing theory until the 1980's, readers can refer to [18] while [23] gives an extensive introduction to key concepts involved in this field.

1.2.1 Single-Server Queues

The most elementary queueing model is the single-server queue where the server works at a constant service rate. In such a model, customers arrive one at a time according to a deterministic process and the times between two consecutive arrivals, referred to as interarrival times, are always constant at $\frac{1}{\lambda}$, where λ is the deterministic arrival rate. Each arriving customer also has an associated service requirement that is deterministic, and which is also independent of the interarrival times. Customers are serviced one at a time by the server and this service cannot be interrupted midway. Finally, customers that have received their full service request leave the system. This above model is commonly referred to as the D/D/1 queue, sometimes followed by the server discipline (the default of which is the First-Come-First-Served 'FCFS' discipline). This notation was introduced by Kendall [8], and is now rather standard throughout the queueing literature. The first D indicates that the interarrival times are deterministic, while the second D indicates that the service requirements are also deterministic. Finally, the last digit 1 indicates that there is a single server. Other commonly used symbols include M for memory-less distributions, such as exponential distributions and G for general distributions.

There exist many possible variations of this basic queueing model including queues with finite capacity. The finite capacity could arise due to there being only a finite number of waiting places in the queue and hence, when all places are occupied, arriving customers would be blocked and typically taken to be lost. This finite capacity could also be interpreted as a bounded waiting time, where customers get impatient if they have to wait too long and decide to leave or renege. Finally, other variants of service disciplines could include a Last-Come-First-Served discipline or a Processor-sharing discipline.

1.2.2 Performance Measures

The performance of a queueing system can be expressed in terms of one or more metrics, the most common of which are queue lengths, waiting times, total system time and throughput. The waiting time is the time spent by customers waiting in the queue, while the total system time represents that total time spent in the system, either waiting or in service. Throughput on the other hand is the long-run average of the number of tasks completed per unit time. Additional performance metrics of interest would then depend on the type of system being analyzed. For example, a performance metric of interest in a HSC setting could be the maximum workload during a certain period (so as not to violate union laws) or the average error rate of tasks completed.

The main performance measure considered in this thesis is the amount of useful throughput of the queue, that is the long-run average of the number of tasks that are executed correctly by the server per unit time. This measure is of interest since purely having a high service rate is useless in reality if every task is executed incorrectly. Such a scenario would actually be akin to not having serviced any tasks at all. Given recent emphasis on the development of error models such as the Drift Diffusion model [35, 2, 20] and the long-standing interest in the speed-accuracy tradeoff, this thesis attempts to merge these mathematicals models into the field of queueing theory and to develop novel optimal policies to maximize the useful throughput of the queue. Note that the key assumption here is that of a binary error model, meaning that every task is either done correctly or not. In addition, this performance measure also implies that the service and error rates of the server are controllable and hence can be optimized. This will be elaborated on more in Chapter 2.

1.3 Queues with State-Dependent Rates

In the queueing model of Section 1.2.1, the server is assumed to work at a constant rate each time it is assigned a task to execute. However, this assumption is not always appropriate as the state of the system may affect the server productivity in some practical situations; Section 1.4 introduces some real-world examples where such behavior may occur. For now, assume the definition of 'state' to generally mean some information summarizing the past history of the system. In our case, we take this to be either the amount of workload in the queue or some external measure of the server's utilization. A more detailed discussion on the notion of a 'state' will be presented in Section 2.3.

To understand the dynamics of a queue with state-dependent arrival and service rates, first imagine a server that is characterized by an observable and controllable state x. The arrival rate of new customers may be influenced by x, for example for queues with finite buffers where the arrival rate can be assumed to instantaneously reduce to zero when the workload in the queue exceed the buffer capacity. Alternatively, a system that consists of n unique queues serviced independently by n unique servers also exhibits such phenomena if the customers constantly jockey and shift to queues of shorter lengths. In this case, a hardworking server that works faster than all the other servers will appear to have an arrival rate that increases with his service rate. Similarly, the service process can depend on the number of customers waiting for service since a server may work faster if the queue is building up or, on the contrary, may get flustered and become less efficient. Nevertheless, despite the more unpredictable nature of such queues, many fundamental derivations of queue stability from typical queueing theory can still be applied and policies developed to maximize the desired performance measures.

1.4 Examples

1.4.1 Production Systems

The classical state-dependent queue that is often quoted in literature can be observed on production floors where the objective is to both maximize throughput and reduce the delay between the time a job enters the system till the time it leaves the system, which is defined as the lead time. Literature such as [17, 37, 13] employed empirical research through simulation and actual implementation of workload-based task allocation policies into actual jobshop settings to investigate the effects of such policies in production systems. In all cases, findings report a substantial reduction in throughput times after the implementation of workload-based policies. For example, in [13], the time taken to achieve the same amount of throughput with the use of workload-based policies was found to decrease by 40% to 50%, compared to when no policy was used.

The possibility of controlling throughput times by controlling the amount of work in the system is due to the relationship of mean throughput against mean inventory as shown in Equation 1.1 [12].

$$TL_m = \frac{I_m}{PE_m} \tag{1.1}$$

where TL_m is the weighted mean delay time, I_m is the mean inventory or queue length and PE_m is the mean performance or mean throughput. Hence, in a production system setting, if one seeks to have a specific flow time for a work center, meaning that it is desired to have an incoming job completed within a certain timeframe, then one must always ensure that a certain mean inventory is maintained. This can be achieved by inputting only so much work during a reference period as is expected to leave the queue in the same period. A change in the total system time of each job can then be achieved by a change in the input (i.e. the load), or by a change in output (i.e. the desired throughput). From the variation of the flow times with the queue length at a certain moment, characteristic production curves can be developed as shown in Figure 1-1.

Figure 1-1 shows the general shape of such curves and clearly demonstrates the change in performance and mean weighted lead time as a function of the mean inventory. One can observe that above a certain inventory value, which is demarcated as the 'critical value', there is no significant increase in performance since there is always enough work to avoid breaks in processing. On the other hand, idle time will occur with increasing frequency below this value resulting in a drop in performance. Similarly, when the inventory is increased beyond the 'critical value', the delay time will increase with increasing inventory since the performance remains almost constant as predicted in Equation 1.1. On the other hand, below this value, the lead time decreases with the inventory. The state in such a system is hence the current inventory

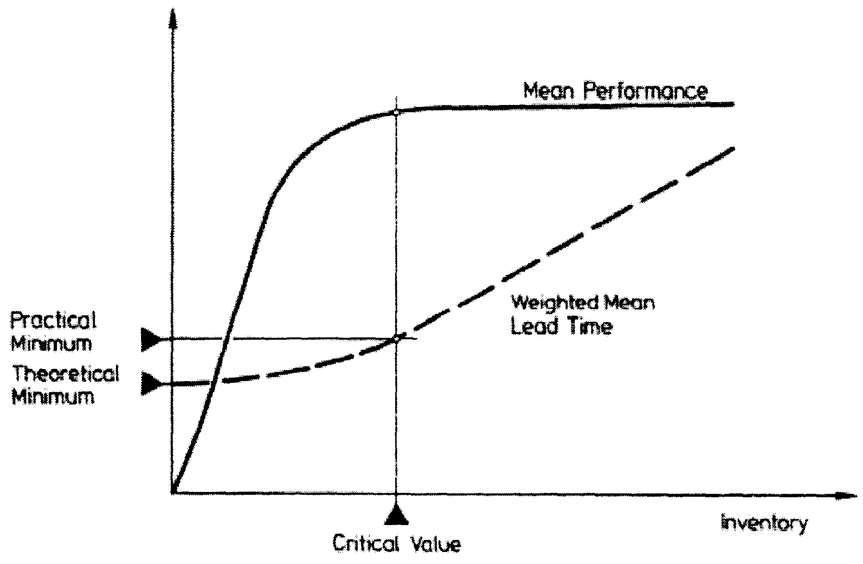


Figure 1-1: Lead time and performance as functions of the inventory [37]

in the production system and the control variable is the workload input or the desired workload output.

1.4.2 Human Supervisory Control Settings

The second example we will examine is concerned with situations where the server is a human being. A typical example would involve a single human operator managing several Unmanned Aerial Vehicles (UAVs) on a mission to patrol and survey a certain airspace for anomalous activity. In such an example, the operator may receive from the UAVs he or she is supervising a constant stream of tasks in the form of videos or images, through which he or she is supposed to detect any suspicious activity. In such cases, the supposition that the server is assumed to work at a constant speed whenever there is any work in the system is almost definitely invalid. There exist, however, two schools of thought on the way the state of the queue could be defined in this case. Nevertheless, both schools of thought draw on the same conceptual law between arousal and performance, which was originally developed by psychologists Robert Yerkes and John Dodson in 1908. The Yerkes-Dodson law models the relationship between arousal and performance as an inverse U-shaped curve, hence implying that

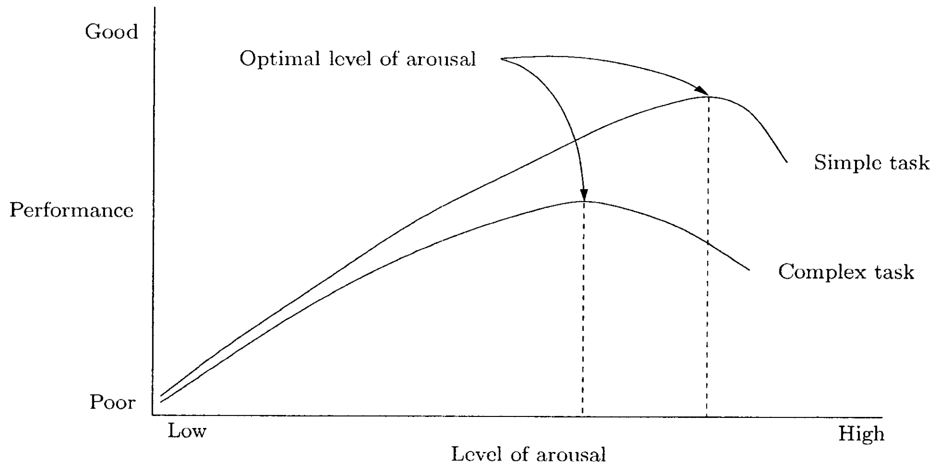


Figure 1-2: Yerkes-Dodson law showing the relationship between arousal and performance [6]

for a given individual and a given type of tasks, there exists an optimal arousal level at which the human's performance has its maximal value. Any increase in arousal would then be considered to be negative work pressure since the human would become less productive from it, while any decrease in arousal would result in the human becoming less efficient. A typical plot characterizing the Yerkes-Dodson law is shown in Figure 1-2.

The first school of thought [6] argues that arousal is caused by 'stressors' in the environments, such as noise, anxiety and fear, but may also involve phenomena such as time pressure. Hence, in a HSC setting, an increased level of arousal could be associated with the psychological effect of 'trying harder', for example a human operator working faster when he notices a buildup of tasks waiting for him to process, compared to when he has no backlog of tasks to work on. Alternatively, if the number of tasks that has accumulated in the operator's backlog increases past a certain value, the operator may experience high stress, which in turn results in a decrease in his ability to process information. For example, the operator might experience *attention narrowing*, or *tunneling*, which in turn results in his direction of all his attention to a single channel of information. For complex tasks involving multiple channels of information, this may have some undesirable side-effects. Moreover, a second consequence of stress is the loss in working-memory, which directly affects human performance. It is interesting to note that this school of thought is somewhat similar to the definition of state as mentioned in Section 1.4.1.

The second school of thought [3, 19] argues that the level of arousal depends on how busy or 'utilized' the human operator has been over the last time period of arbitrary length. In [3], the authors attempted to investigate this claim by developing a software tool to allow human operators to supervise a team of unmanned vehicles in a simulated operational environment. Operators were engaged in a variety of tasks including waypoint selection, target assignment and visual classification. The findings of the experiment showed that there existed a U-shaped relationship between the level of situational awareness of an operator and the operator's utilization factor. Situational awareness was measured in terms of the reaction time of an operator to respond to a threat area once it intersected with the path of a UAV and is what we could classify as the human's performance. On the other hand, the utilization factor was calculated as the proportion of time over fixed intervals of 150 seconds that the operator was busy interacting with the display during the experiment, which is very much akin to the level of arousal of the human over that fixed interval of time.

The authors in [19] take this one step further and propose and verify (using the same set of experimental data) a first-order differential equation to model the level of arousal of the human operator based on whether or not the human operator is busy or idle. Figure 1-3 shows the validity of the first-order model in deducing a measure of the level of arousal of the human operator, since the trend observed in the figure is validated by our expectation from the Yerkes-Dodson law. Note that in the figure, the circles represent the average of the data in each 5% utilization factor bin, the black solid line is the weighted least-squares quadratic approximant, the numbers indicate the number of samples in the bin, and the error bars show the $1-\sigma$ confidence intervals.

1.4.3 Traffic Flow Control

A third example of state-dependent queues can be seen on a day-to-day basis on the roads we travel on. Congestion modeling has been a topic under continuous

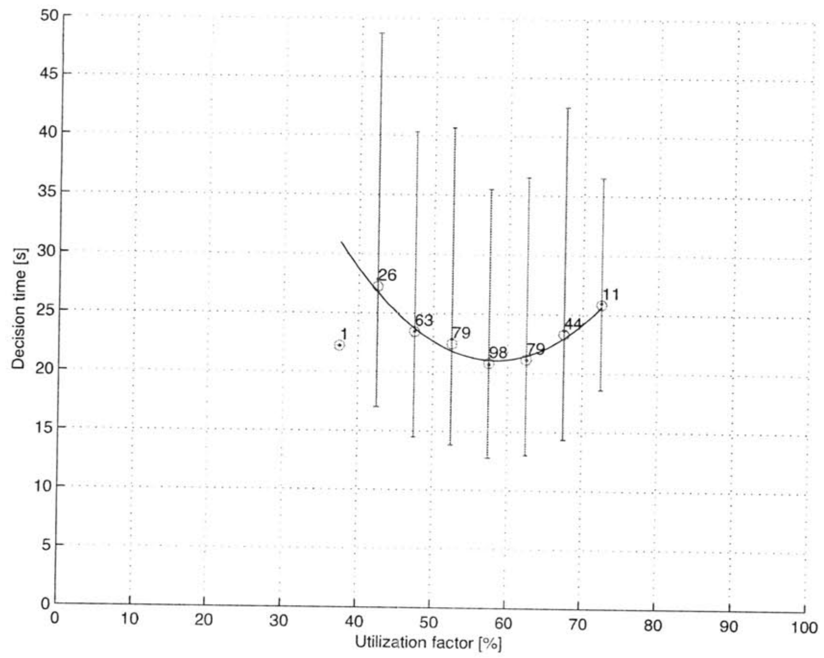


Figure 1-3: Plot of decision times in classification tasks verifying the Yerkes-Dodson law [19]

consideration in traffic flow theory since Greenshields' study of traffic capacity in 1933 [4]. Greenshields postulated a linear relationship between speed and traffic density, which proposes that the more vehicles there are on a road, the slower their velocity would be. This in turn converts into a parabolic relation between speed and traffic flow, where traffic flow is calculated to be the product of the traffic density with the speed. A fundamental traffic diagram is then a diagram that gives the relation between any 2 of the following 3 variables: the speed, the traffic flow and the density of vehicles on a road. These diagrams are unique to each type of road network and such a diagram has been obtained both theoretically and by simulation for one road [29]. Figure 1-4 shows the typical fundamental traffic flow diagram under a classical Lighthill-Whitham-Richards (LWR) model. The basic idea of this theory is that the maximum flow rate q_{cap} associated with the maximum point (o_{cr}, q_{cap}) in the figure determines the free flow capacity at a bottleneck. More information about this theory can be found in [5].

Given the trend of a typical flow-density fundamental traffic diagram, and in an

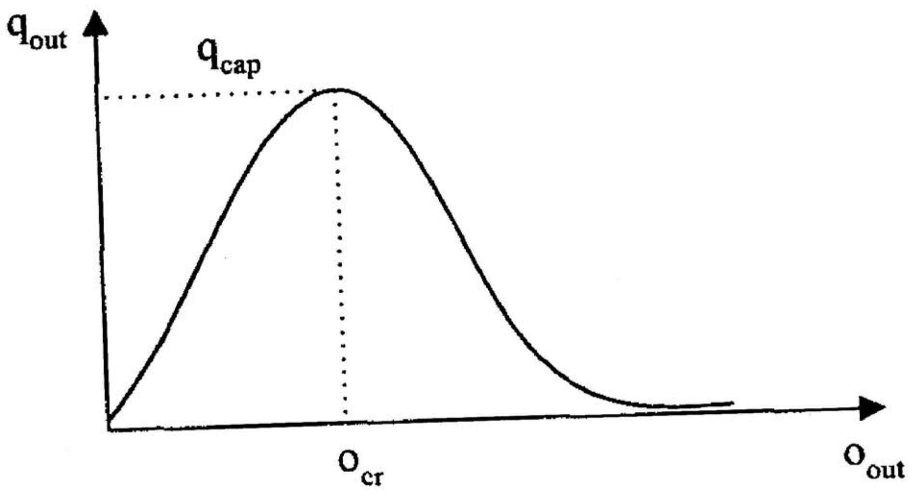
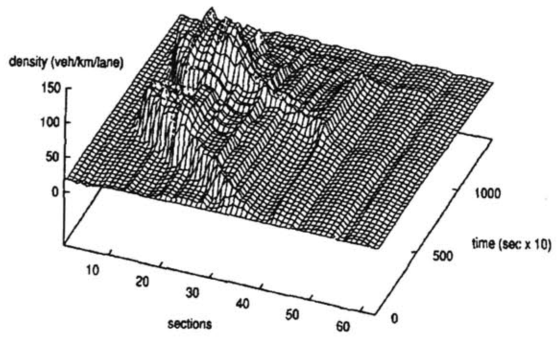


Figure 1-4: Fundamental traffic diagram [5]

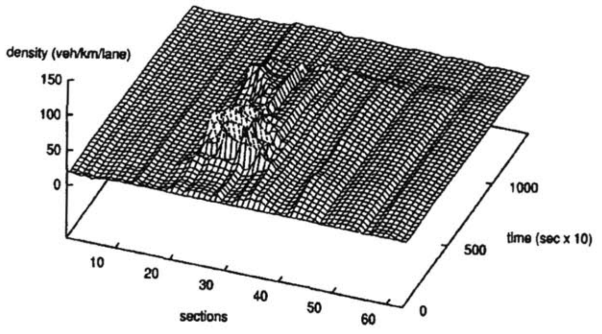
effort to relieve peak hour congestion on freeways, various ramp metering algorithms have been employed to regulate the inputs to freeways from entry ramps. In fact, these has been in use for over 30 years and are presently employed in a number of urban areas in North America [7]. There exist several types of ramp metering strategies including fixed-time strategies that are derived offline based on constant historical demands and reactive strategies that aim to keep the freeway traffic conditions close to pre-specified set values based on real-time measurements. The *demand-capacity strategy* [10] for example, which belongs to the latter category, specifies the ramp input rates at a local section of a highway as shown in Equation 1.2.

$$r(k) = \begin{cases} q_{cap} - q_{in}(k-1) & \text{if } o_{out}(k) \leq o_{cr} \\ r_{min} & \text{else} \end{cases} \tag{1.2}$$

where q_{cap} is the freeway capacity downstream of the ramp, q_{in} is the freeway flow measurement upstream of the ramp, o_{out} is the freeway occupancy measurement downstream of the ramp, o_{cr} is the critical occupancy at the same location at which the freeway flow is maximum and r_{min} is a pre-specified minimum ramp flow value. This policy hence attempts to add to the measured upstream flow as much ramp flow as needed to reach the downstream freeway capacity. However, should the downstream



(a) No control



(b) With control

Figure 1-5: Comparison of flow density without and with ramp metering [26]

occupancy exceed a critical value when congestion occurs, then the ramp flow is reduced to a minimum flow to avoid and dissipate the congestion.

The benefits of such strategies can be seen from simulation plots in Figure 1.4.3 comparing the flow density on a multi-lane freeway without and with control [26].

1.5 Thesis Overview

The main focus of this thesis is the development of optimal policies to maximize the useful throughput of a state-dependent queue with deterministic arrival rates. The service times and error rates of the server are assumed to be dependent on the state of the server, and a first-order differential equation is proposed to model the evolution of the server state. In addition, this thesis summarizes the results and lessons learnt from the proof-of-concept experiments, which were designed and implemented specifically to test the feasibility of applying the optimal policy in a real-world setting. This was done using a simple computer interface with subjects recruited from campus. Some basic ideas for estimation of server parameters and a variety of other related case studies are also mentioned with slight detail in this thesis.

Chapter 2 summarizes existing literature and work that have been done in the field of state-dependent queues and in the area of optimal control of queues. In addition, various methods that have been proposed in literature to derive a measure of the server state are discussed, particularly in the field of Human Factors. Chapter 3 analyzes the D/G/1 queue with state-dependent and deterministic service rates and proves the optimality of a one-task threshold policy in maximizing the useful throughput of the queue. Slightly varying threshold policies are then proposed for cases where the objective function either includes costs associated with switching the server on-and-off and/or costs associated with task waiting times. Chapter 4 extends the work done in Chapter 3 to the case where the server is assumed to behave stochastically and demonstrates the infeasibility of the policy from Chapter 3 when extended to a stochastic server. Chapters 5 and 6 introduce the proof-of-concept experiment that was designed to test the real-world feasibility of the proposed policy on actual human subjects and document the results and lessons learnt from the implementation respectively. Finally, Chapter 7 summarizes our findings and conclusions from this research.

Chapter 2

Background and Previous Work

Queueing theory is a field with a long history. In this chapter, we examine some previous work that has been done in the field of both optimal control of queues and in the field of state-dependent queues. In addition, various methods that have been proposed in literature to derive a measure of the server state are discussed, particularly in the field of Human Factors.

2.1 Background on Optimal Control of Queues

The purpose of optimal control of queueing systems is to determine when and how to change arrival or service rates to optimize some objective function [24]. This typically means determining the queue levels at which service should start or stop. Much research has been focused on finding optimal operating policies for specific settings, which turn the server on and off in ways that result in the lowest long-run cost. Possible policies can be classified into several categories, the most common classification of which is that of stationary and non-stationary policies. Stationary policies are policies that always demand the same action whenever the system is in a given state. The earliest of this model was used by Romani [14] in determining a policy for the optimal number of servers to employ. Further elaboration on possible policies in each of these categories will be given for our specific problem setting in Section 3.2.

A typical problem formulation along with the usual optimal policy candidates is presented in [1]. In the article, the authors consider a queueing system without any state dependencies and with a single server who has the option of leaving for a vacation period of random length as soon as the system empties out, that is whenever the number of customers in the queue is zero. In reality, we could imagine the server to leave to perhaps perform other activities, but from the customers' point of view, the server is considered to be taking a vacation. The queue dynamics require that customers are served one at a time with i.i.d service times, while the vacation times are also considered to be i.i.d. Arrival, service and vacation times are also assumed to be independent of each other. The cost function associated with this queueing system consists of a holding cost rate specified by a general non-decreasing function of the queue size, fixed costs for initiating and terminating service, and variable operating cost incurred for each unit of time that the server is in operation, which also depends on whether it is serving a customer or remaining idle. This is a very common set-up of a problem from which an optimal policy is desired and in their article, the authors prove that under varying conditions, the following three threshold policies are optimal: the N-threshold policy, the D-threshold policy and the T-threshold policy. This section will hence give a short summary on each of these policies to give readers a clearer picture on optimal policies for queues without any state-dependencies.

2.1.1 N-, D-, T-, and Other Threshold Policies

Of particular interest among stationary policies are the threshold policies, which turn the server on only when the queue size is equal to or larger than a given value, and turn the server off when the queue is empty. Optimality of threshold policies have been shown in much literature and readers can refer to [33, 15] for more details. We will instead focus on the three different threshold policies that are most commonly referred to in literature: the N-policy [27], the D-policy [21] and the T-policy [9].

The N-policy is a stationary policy that turns the server on when the total number of tasks in the queue reaches the value N and turns the server off when the system becomes empty. The rationale behind such a policy is to determine a tradeoff between the average wait time of a task in the queue and the costs incurred in turning the server on and off, versus the cost of leaving the server active all the time. Hence, in order to minimize cost, the server is only turned on when there is sufficient number of tasks in the queue to service in one continuous manner and turned off once there are no more tasks to service. However, cost considerations of keeping tasks waiting in the queue are also considered. The D-policy is a variant of the N-policy and takes into account the service times of the waiting customers. Hence, a D-policy is also a stationary policy that switches an idle server on when the workload reaches or exceeds the level D and switches it back off when the system becomes empty. Using the same cost function as before, the authors in [22] show that the D-policy is superior to the N-policy for exponential service time distributions.

One limitation of the above two mentioned policies, however, is that in order to employ them, the server must continuously monitor the queue for an arrival when the server is idle. In the case that this is not possible, for example due to cost and efficiency consideration, then the T-policy can be used. A stationary T-policy is one where the server scans the queue T time units after the end of the last busy period to determine if customers are now present in the queue. The presence of customers begins a busy period and the server is active until the queue is emptied. However, if no customers are present in the queue, then the next scan is made after T time units again.

There exist a multitude of other variants of the above 3 policies and readers are advised to refer to [24] for more details.

2.2 Background on Optimal Control of State-Dependent Queues

Research on state-dependent queues differs from those without state-dependence due to the dynamic nature of the queue behavior. For example, assuming that the state of the queue is defined to be the amount of tasks waiting in the queue, then employing a high service rate would be beneficial. However, as the queue empties out, maintaining the high service rate might be costly and hence the service rate should optimally be lowered gradually as the queue gradually empties out. Similarly, if the state of the queue is defined to be how much work the server has performed in the past, then the optimal policy might be to maintain an optimal workload for the server so as to increase overall efficiency, even though the short-sighted policy might be to overwork the server so as to clear the backlog.

The first investigation into optimal control policies for state-dependent queues was done in [28], which assumed that the service rate of a server could be varied at any time and is under the control of the decision-maker. The service capacities or speeds of the server are denoted by $\mu_0, \mu_1, \cdots, \mu_k, \cdots$, where $\mu_{k+1} > \mu_k$ and $\mu_0 = 0$. The authors found the optimal policy to be one where the service capacity is increased from μ_{k-1} to μ_k when the queue length reaches a value R_k from below. On the other hand, when the queue length drops to S_k from above, then the server speed is reduced from μ_{k+1} to μ_k. The vectors R and S are represented by $\{R_1, R_2, \cdots, R_k, \cdots\}$ and $\{S_1, S_2, \cdots, S_k, \cdots\}$ respectively, where $R_{k+1} > R_k$, $S_{k+1} > S_k$ and $R_{k+1} > S_k$. Such a policy can be said to be one that demonstrates *hysteresis*.

Much work on state-dependent queues has also been done in [31, 32, 30] where the authors once again considered queues with service rates that are dependent on the amount of workload in the queue. The service rate was also assumed to be first increasing then decreasing as a function of the amount of work and the admission of work into the system was controlled by a policy for accepting or rejecting jobs, depending on the state of the system. In [32], the authors found that a threshold policy is optimal in maximizing the long-run throughput within the class of stationary deterministic policies. The authors, however, did not include in their cost function costs for rejecting jobs as their only objective was to maximize throughput.

To the best of the author's knowledge, however, no results are available yet on state-dependent queues where the state is dependent on the amount of work the server has done. This will hence form the main aim of this thesis. The following section provides some discussion on the measures of state and aims to be an elaboration of what was discussed earlier in Chapter 1.

2.3 Measures of Server State

There exist many methods to measure the state of a server based on the amount of work the server has done. These methods can be broadly categorized into 2 classes: Objective measures and Subjective measures. Methods belonging to the former category are useful in deriving mathematical proofs to obtain optimal policies and can work 'behind-the-scenes' to monitor the server state without seeking any direct input from the server. There are, however, many concerns over the accuracy and correctness of such models, not to mention that the need to differentiate human behavior by utilizing different model parameters is another problem in inference in itself. The latter category consists many of models that require the server to provide some form of feedback to indicate how busy he thinks he is. This method is very popular in the field of Human Factors, though results are highly subjective and unwanted correlations hard to avoid.

2.3.1 Objective Measures

One method of measuring the server state can be done by finding the proportion of time over fixed intervals of t seconds that the server was busy [3]. A measure of 'busyness' could be taken to be when the server interacts with a user interface or simply when the server has been assigned a task. Another method of measuring the server state was proposed in [19], which proposed a first-order differential equation as shown in Equation 2.1.

$$\dot{x}(t) = \frac{b(t) - x(t)}{\tau}, \qquad x(0) = x_0, \tag{2.1}$$

where τ is a time constant that determines the extent to which past utilization affects the current state of the server, and $x_0 \in (0, 1)$ is the initial condition. Note that the dynamics described by Equation (2.1) is such that, for any $\tau > 0$, $x_0 \in (0, 1)$ implies

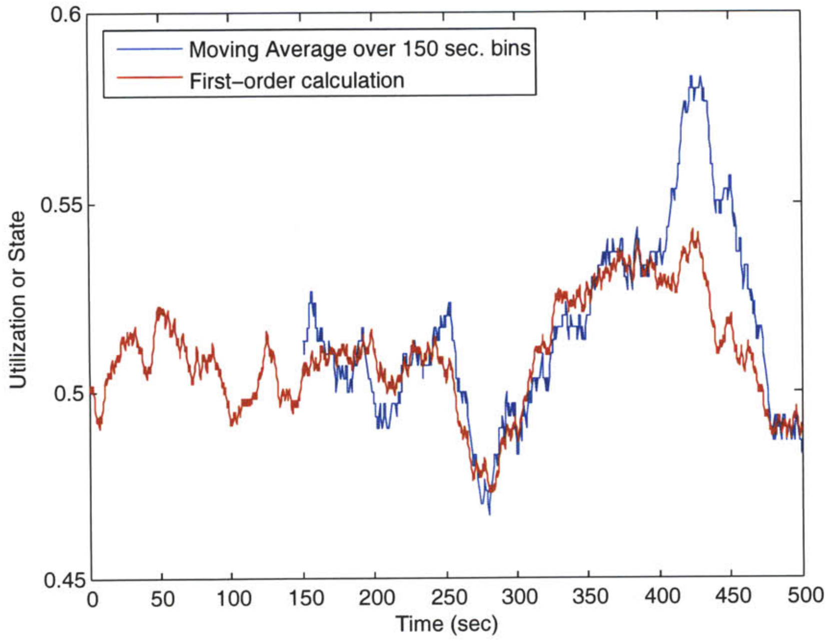


Figure 2-1: Comparison between moving average calculation and first-order model calculation of state

that $x(t) \in (0, 1)$ for all $t \geq 0$.

Figure 2-1 compares the output from these two methods for a fixed interval of 150 seconds. The blue line represents the moving average calculation of the percent busy time over time intervals of 150 seconds and hence only starts after 150 seconds has passed, while the red line represents the state calculation using Equation 2.1. The latter method was calculated using $\tau = 150$ and an initial $x_0 = 0.5$. We notice the similarity in trend between the two methods from Figure 2-1, though apart from that, not much else is similar. An investigation into the sensitivity of the latter method with respect to x_0 is also shown in Figure 2-2, which compares the state progression for the same set of data but using different initial conditions. We note that the state measurement using different starting conditions eventually converges, though the amount of time taken for the state to converge could be a hindrance during actual implementation.

Furthermore, the latter model also has the propensity to distinguish between different types of server behaviors, for example, in the case of the servers being human, some humans could perform better under stress compared to others. Hence, the

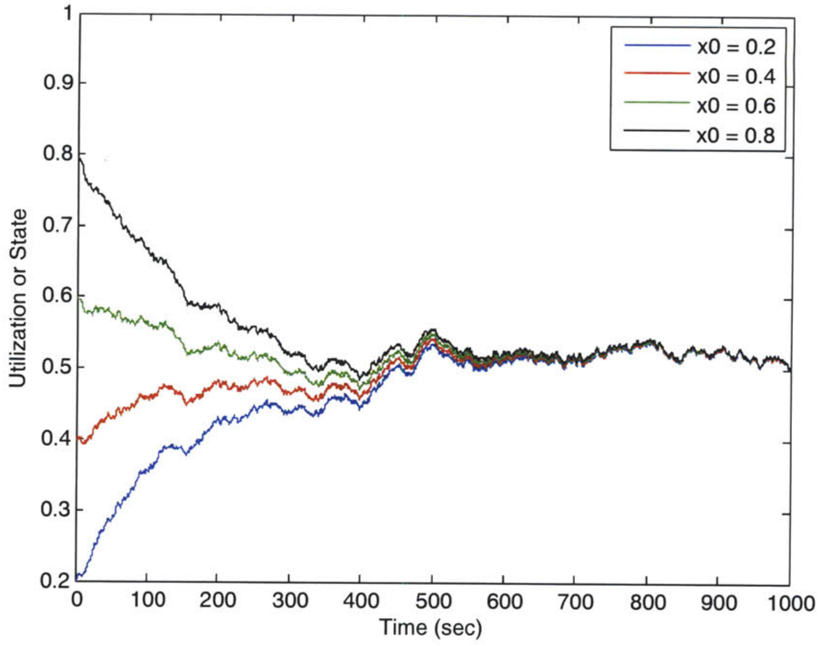


Figure 2-2: Comparison between state progression for first-order model under different initial conditions

latter model could be able to distinguish such differences through the use of different τ values. Based on our definition, higher τ values could indicate a longer working memory and possibly less sensitivity to short-term changes in busyness. Hence, from Figure 2-3, we notice that as the value of τ increases, the server state displays less sudden jumps over time, though the general trend of the state for all values of τ is the same. For the remainder of this thesis, we will be employing the latter mathematical model in Equation 2.1 to measure the state of the server.

2.3.2 Subjective Measures

Subjective workload measures are pre-dominantly associated with the field of Human Factors, and hence are typically used only when the server is a human. Such measures are popular for use due to the ease of administration and the low cost of implementation. Some commonly used measures include the NASA-TLX and SWAT. Nevertheless, such measures also bring about their own problems, including a high dependence on short term memory and the subjective nature of the response illicited from the human due to different people's interpretation.

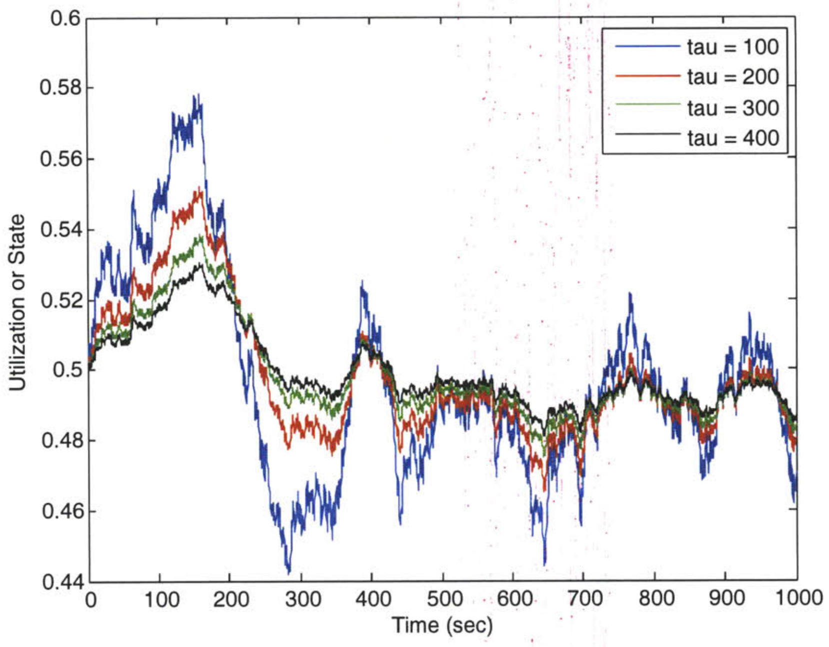
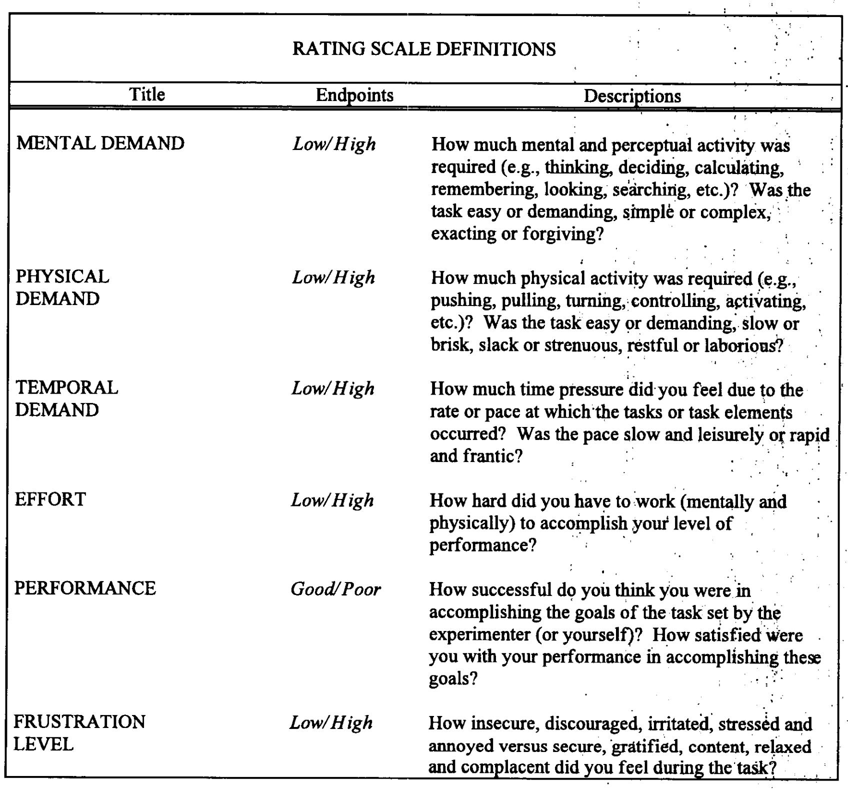


Figure 2-3: Comparison between state progression for first-order model using different τ values

The NASA-TLX or Task Load Index is a subjective workload assessment tool that allows users to perform workload assessments on operators working with various human-machine systems. The NASA-TLX is a multi-dimensional rating procedure that derives an overall workload score based on a weighted average of ratings on six subcategories: Mental Demand, Physical Demand, Temporal Demand, Own Performance, Effort and Frustration. Despite the popularity of this method however, this remains a very tedious process to obtain a workload measure and the problem of individual differences is not addressed. Figure 2-4 shows the rating scale defitions for the NASA-TLX.

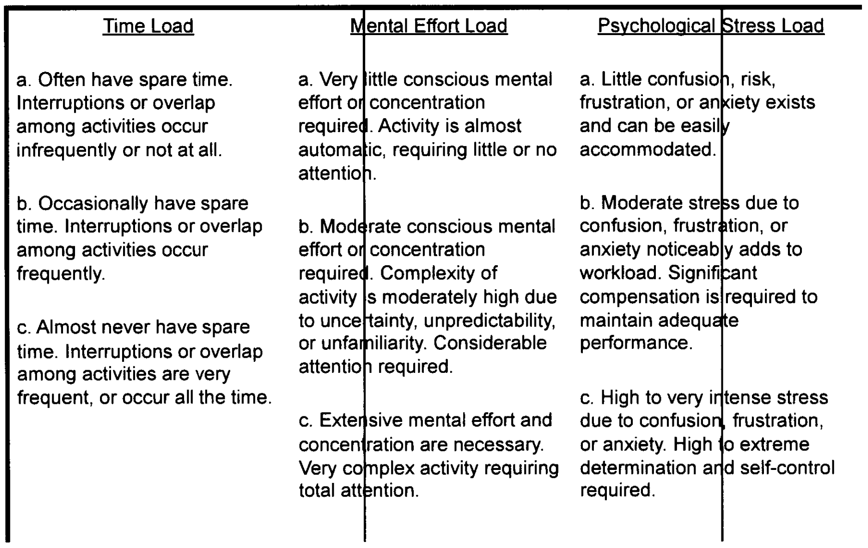
SWAT, otherwise known as the Subjective Workload Assessment Technique, is an alternative to the NASA-TLX and provides an easily administered subjective scaling method that is commonly used in flight cockpits to quantify the workload associated with various activities. SWAT postulates a multi-dimensional model of workload comprising three, three-point dimensions: Time Load, Mental Effort Load and Psychological Stress Load. Problems associated with this method, however, include correlations between the three dimensions, the lack of diagnosticity and low



RATING SCALE DEFINITIONS

Title	Endpoints	Descriptions
MENTAL DEMAND	*Low/High*	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
PHYSICAL DEMAND	*Low/High*	How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	*Low/High*	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
EFFORT	*Low/High*	How hard did you have to work (mentally and physically) to accomplish your level of performance?
PERFORMANCE	*Good/Poor*	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
FRUSTRATION LEVEL	*Low/High*	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Figure 2-4: Example of rating scale definitions for NASA-TLX workload measure



Time Load	Mental Effort Load	Psychological Stress Load
a. Often have spare time. Interruptions or overlap among activities occur infrequently or not at all.	a. Very little conscious mental effort or concentration required. Activity is almost automatic, requiring little or no attention.	a. Little confusion, risk, frustration, or anxiety exists and can be easily accommodated.
b. Occasionally have spare time. Interruptions or overlap among activities occur frequently.	b. Moderate conscious mental effort or concentration required. Complexity of activity is moderately high due to uncertainty, unpredictability, or unfamiliarity. Considerable attention required.	b. Moderate stress due to confusion, frustration, or anxiety noticeably adds to workload. Significant compensation is required to maintain adequate performance.
c. Almost never have spare time. Interruptions or overlap among activities are very frequent, or occur all the time.	c. Extensive mental effort and concentration are necessary. Very complex activity requiring total attention.	c. High to very intense stress due to confusion, frustration, or anxiety. High to extreme determination and self-control required.

Figure 2-5: Example of rating scale definitions for SWAT workload measure

mental workload issues. Figure 2-5 shows an example of the SWAT scale.

2.4 Human Supervisory Control

Unmanned Aerial Vehicles (UAVs) are commonly used for wide-area surveillance and low-altitude UAVs in such a mission must provide coverage of a region and investigate events of interest, possibly with the assistance of a human operator, as they manifest themselves [19]. In particular, we are concerned with cases in which close-range information is required on targets detected by high-altitude aircraft, spacecraft, or ground spotters, after which the UAVs are then sent to the locations to service the target under direct operator supervision. Servicing of targets could include tasks like gathering on-site information, target classification, or engagement as shown in Figure 2-6.

In addition, UAV operators in the future will be expected to be supervisors of multiple UAVs. Hence, workload has become a major factor in determining the number of UAVs a single human operator can effectively control, and therefore the effect of workload on performance has been an important relation to study over the years. Within the realm of Human Factors, this has led to literature investigating such a relation and proposition of formulae for predicting the number of agents a

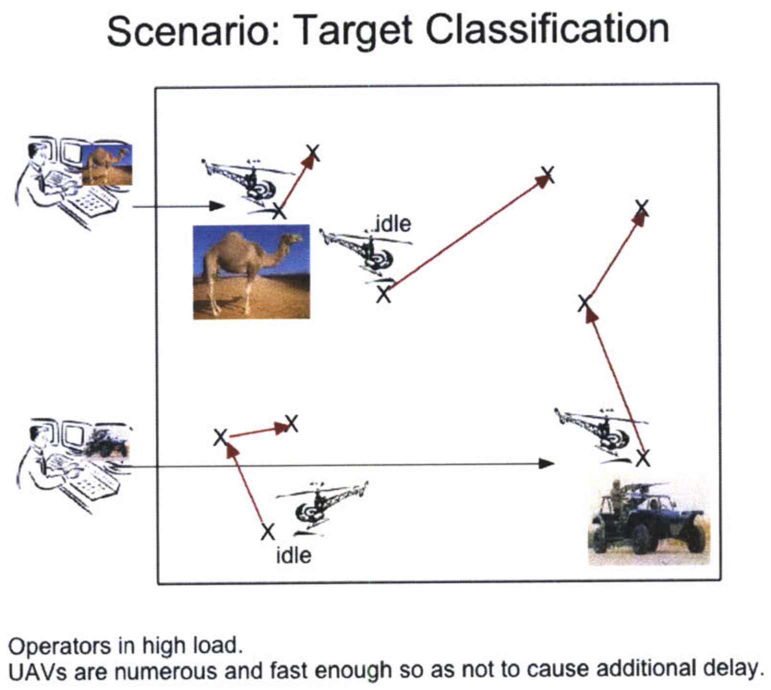

Scenario: Target Classification
idle
idle
Operators in high load.
UAVs are numerous and fast enough so as not to cause additional delay.

Figure 2-6: Target classification scenario involving cooperation between human operators and UAVs [19]

human can effectively control [25]. This formula was based on the time an agent can be neglected and the time it takes to interact with the agent to raise its performance to an acceptable level. The formula, however, did not consider how the operator's workload could affect his performance in interacting with the agent. Some measures of performance could include the time taken to interact with the agent or the efficiency of the operator when interacting with the agent. We will hence attempt to address these issues in this thesis.

Moreover, given the hypothetical mission profile, two key performance measures are the time it takes for a human operator to service a target (for example the time taken for the operator to classify the target to either be a hostile or a friendly) and the probability that the human operator makes a mistake (for example classifying a hostile to be a friendly or vice versa). Given the findings in [19], we hypothesize that the variation of service times with our measure of state varies according to the Yerkes-Dodson law, though we will later also discover this to be inconsequential to the derivation of the optimal policy. The variation of the error probability of the human operator with respect to his state is however an area that has not been discussed previously.

For this thesis, we hypothesize that the error rates follow the Yerkes-Dodson law as well, meaning that the error rates decrease up until a certain optimal state before increasing again. In this section, however, we will introduce the types of error models that have been proposed in literature in recent years.

2.4.1 Speed-Accuracy Tradeoff and Error Models

The speed-accuracy tradeoff has been a very widely studied phenomenon and the range over which one can obtain substantial speed-accuracy tradeoff varies from 150 milliseconds in some very simple perceptual tasks to 1000 milliseconds in some recognition memory tasks and probably even longer in more complex cognitive tasks. There exist several types of speed-accuracy tradeoff theories, including the fast guess theory, the discrete process theory with a distribution of finishing times, and the continuous strength-integration theory [2]. For example, according to the fast guess theory [34], a subject typically responds with either a random guess with short latency or with a stimulus-controlled response at considerably longer latency. More information on these theories can be found in [2]. The types of error models, however, that we will be concerned with are binary error models, where the subject either gets it correct or wrong, with no in-between.

For example, consider a scenario where a human has to decide on one of two hypotheses, H_0 and H_1, based on the amount of stimulus or evidence collected. There currently exist 2 models that are popular to model the speed-accuracy tradeoff in such binary decision models. The first is the Pew's model [35, 20] which states that the probability of selecting H_1 given that H_1 is true at a given time $t \in \mathbb{R}_{\geq 0}$ where t is the time from the start of the consideration of the hypotheses is given by Equation 2.2.

$$P(\text{say } H_1|H_1, t) = \frac{p_0}{1 + e^{-(at-b)}} \tag{2.2}$$

where $p_0 \in [0, 1]$ and $a, b \in \mathbb{R}$ are some parameters which depend on the human

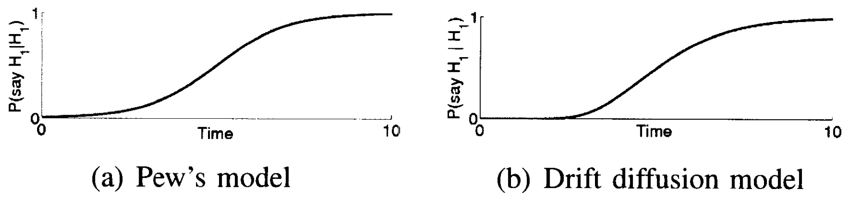


Figure 2-7: Speed-Accuracy Tradeoff Operating Characteristic for the Pew's Model and the Drift Diffusion Model [36]

operator. The second is the Drift diffusion model which states that given the hypothesis H_1 is true, the evolution of the evidence for decision is modeled as a drift-diffusion process given by Equation 2.3.

$$P(\text{say } H_1|H_1,t) = \frac{1}{\sqrt{2\pi\sigma^2 t}} \int_{\eta}^{\infty} e^{\frac{-(\Lambda-\beta t)^2}{2\sigma^2 t}} d\Lambda \tag{2.3}$$

where $\beta > 0$ is the drift rate, σ is the diffusion rate, η is the decision threshold, and Λ is the evidence at time t. A plot of the conditional probabilities for both models is shown in Figure 2-7.

Chapter 3

Policies for Servers with Deterministic Performance

In this chapter, we study the stability problem of a dynamical queue with deterministic state-dependent service times and error rates. The evolution of the server state, and hence the service times and error rates rendered by it are governed by its utilization history. The first-order differential equation proposed in Equation 2.1 is used to model the evolution of the server state while the service times and error rates are hypothesized, as mentioned in Chapter 2, to be related to the server state by a continuous positive convex function. The objective of this queueing system is to maximize the amount of useful throughput of the queue, where useful throughput is defined to be the number of tasks executed correctly by the server per unit time. The server is assumed to be attempting to perform optimally and hence allocating his time resource in such a way that the objective is maximized. Finally, we also examine cases with switching and holding costs, the details of which will become clearer later in this chapter.

In this problem, identical and independent tasks arrive at a deterministic rate and need to be serviced by the single server in the order in which they arrive. We are interested in designing an operating control policy that minimizes our objective function, which we define to be some linear combination of the number of correct answers and the number of tasks answered. The optimal control policy is hypothesized to be a threshold policy with a threshold value that would be somewhere in between value that results in the minimum error rate and the threshold value associated with purely maximizing throughput.

3.1 Problem Formulation

Consider a single-server queuing model with tasks arriving at a deterministic rate of λ. Tasks are identical and independent of each other and we assume that tasks must be serviced in a first-come-first-served manner. The service times of the server are assumed to be state-dependent and **deterministic** (that is the time taken to service a task at a certain state is always fixed). The dynamical model for the server that determines the service times for each task is a simple first-order model as shown in Equation 2.1 and is redefined here for completeness.

Let $x(t)$ be the server state at time t. Let $b : \mathbb{R} \to \{0, 1\}$ be an indicator function such that $b(t)$ is 1 if the server is busy at time t and 0 otherwise. The evolution of $x(t)$ is then governed by the simple first order model:

$$\dot{x}(t) = \frac{b(t) - x(t)}{\tau}, \qquad x(0) = x_0, \tag{3.1}$$

where τ is a time constant that determines the extent to which past utilization affects the current state of the server, and $x_0 \in (0, 1)$ is the initial condition. Note that the dynamics described by Equation (3.1) is such that, for any $\tau > 0$, $x_0 \in (0, 1)$ implies that $x(t) \in (0, 1)$ for all $t \geq 0$.

The service times are related to the state $x(t)$ through a map $\mathcal{S} : (0, 1) \to \mathbb{R}_+$. If a task is allocated to the server at state x, then the service time rendered by the server on that task is $\mathcal{S}(x)$. Since the controller cannot interfere the server while it is servicing a task, the only way in which it can control the server state is by scheduling the beginning of service of tasks after their arrival. Such controllers are called admission controllers and will be formally characterized later on. Furthermore, we also assume that $\mathcal{S}(x)$ is positive valued, continuous, **convex**, is bounded within the interval $x \in (0, 1)$, and that $\mathcal{S}_{\min} := \min\{\mathcal{S}(x) \mid x \in (0, 1)\}$, and $\mathcal{S}_{\max} := \max\{\mathcal{S}(0^+), \mathcal{S}(1^-)\}$.

Similarly, the error probabilities are related to the state $x(t)$ through a map $\mathcal{E} : (0,1) \to [0,1]$. If a task is allocated to the server at state x, then the probability of the server getting the task wrong is $\mathcal{E}(x)$. Similar to the determination of the service time, we also assume that $\mathcal{E}(x)$ is positive valued, continuous and **convex**. Due to its definition, we know that this value is definitely bounded within the interval $x \in (0,1)$, and that $\mathcal{E}_{\min} := \min\{\mathcal{E}(x) \mid x \in (0,1)\}$, and $\mathcal{E}_{\max} := \max\{\mathcal{E}(0^+), \mathcal{E}(1^-)\}$.

Given a dynamical queue with server dynamics shown in Equation (3.1), we now examine the types of admission control policies possible. An admission controller u acts like an on-off switch at the entrance of the queue whereby $u(t) \in \{\text{ON}, \text{OFF}\}$ for all $t \geq 0$, and an outstanding task is assigned to the server if and only if the server is idle, i.e., when it is not servicing a task, *and* when $u = \text{ON}$. Otherwise, the task is simply left in the queue until $u = \text{ON}$ (Note that this is unlike the admission controllers mentioned in [31, 32, 30] which turn away tasks when $u = \text{OFF}$). Let $\mathcal{U}$ be the set of all such admission control policies, and hence allow $\mathcal{U}$ to be quite general in the sense that it includes control policies that are functions of λ, $\mathcal{S}$, x, etc.

3.1.1 Problem Statement

The problem statement is as follows: For a given $\tau > 0$, let $f_u(\tau)$ be the objective cost that we want to maximize at time t under admission control policy $u \in \mathcal{U}$ when the task arrival rate is λ and when the server state at time $t = 0$ is x_0. In order to maximize the useful throughput of the queue, which we define to be the number of tasks executed correctly per unit time, we seek to maximize $f_u(\tau)$, which is in turn defined to be:

$$f_u(\tau) := \lim_{t \to \infty} \frac{B_u(0,t,\tau)}{t} \tag{3.2}$$

assuming the limit to exist. Here $B_u(0,t,\tau)$ denotes the number of tasks completed **correctly** during $[0,t]$ under policy u. Hence, this results in $f_u(\tau)$ defining the total useful throughput over the timeframe $[0,t]$. Note that a policy u^* is said to be (strictly) optimal if $f_{u^*}(\tau) > f_u(\tau)$ $(f_{u^*}(\tau) \geq f_u(\tau)$.

Note that there exist multiple ways to calculate $B_u(0,t,\tau)$ and two such possibilities will be introduced here. The first possibility proposes that $B_u(0,t,\tau)$ is a nonlinear relationship between the average probability of getting a task correct in the timeframe $[0,t]$, $\overline{C_u(t,\tau)}$ and the number of tasks that the server attempted during the timeframe $[0,t]$.

$$B_u(0,t,\tau) := t \cdot \overline{C_u(t,\tau)}^a \cdot \lambda_{max,u}^b$$

In the above expression, a , $b \in \{0,1\}$. This means that if the server is purely maximizing his throughput and not his useful throughput, $a = 0$ and $b = 1$. On the other hand, if the server is purely trying to maximize the percentage of questions he answers correctly, then $a = 1$ and $b = 0$. The second possibility proposes that $B_u(0,t,\tau)$ is a linear combination of $\overline{C_u(t,\tau)}$ and the number of tasks that the server attempted during the timeframe $[0,t]$.

$$B_u(0,t,\tau) := t \cdot (c\overline{C_u(t,\tau)} + (1-c)\lambda_{max,u}(t,\tau))$$

where c is a constant, $c \in [0,1]$ that is uniquely determined by the server and depends on the way the server chooses to place value on getting a task done right versus getting as many tasks done.

In the two equations above, note that we define the maximum stabilizable arrival rate for policy u as:

$$\lambda_{\max}(t,\tau) = \sup\left\{\lambda \mid \exists x_0 \in (0,1), q_0 \in \mathbb{N} \text{ s.t.} \limsup_{t\to+\infty} q_u(t,\tau,\lambda,x_0,q_0) < +\infty\right\}.$$

where $q_u(t,\tau,\lambda,x_0,n_0)$ represents the queue length (outstanding tasks yet to be be serviced in the queue) at time t under the admission control policy $u \in \mathcal{U}$ and when the task arrival rate is λ. x_0 and q_0 are defined to be the initial server state and initial size of the queue respectively at $t = 0$. Note that in the case that we opt to purely maximize throughput, this maximally stabilizable arrival rate will indicate that as long as the server keeps the queue stable, all tasks will eventually be completed and the throughput would be the highest possible. In our analysis below, we will pre-dominantly be dealing with the latter definition of $B_u(0, t, \tau)$ due to the ease in analyzing a linear objective function though some analysis will be done for the former definition as well.

3.2 Determining the Optimal Policy

First, assume that the decision to assign or not assign a task to the server occurs only when the server is idle. For a given policy u, we use $u(x) = 1$ to denote that the server is assigned a task when the state of the server equals x while $u(x) = 0$ denotes that the server is allowed to remain idle until the next decision on whether or not to assign a task to the server. Note that no other control actions are available for such a queueing system since we assume that service cannot be interrupted midway through a task. In addition, there only exist a single stream of tasks arriving at a deterministic rate and no other secondary tasks can be added. From these assumptions, we can hypothesize several types of policies that could be applied to such a system, drawing on the background of optimal control in queues discussed in Chapter 2. These policies can primarily be classified into 3 groups: *Stationary*, *Non-stationary* and *Greedy.*

- **Stationary Policies**:
  - Single x-Threshold Policies: Task assigned when x is below or equal to a certain threshold value and not assigned otherwise
  - Dual Switching Threshold Policies: Task(s) assigned when x is below or equal to a certain lower threshold value and continuously assigned thereafter until x exceeds an upper threshold value. Such policies are known for their hysteresis property.
- **Non-stationary Policies:**
  - n-Task Policy where n tasks are done by the server after which the server remains idle for a certain amount of time before accepting tasks again. This results in different server behavior for a certain server state. A randomized

policy is a variant of this whereby the amount of time that the server remains idle is a random value.

- **Greedy**
  - Server accepts a new task as long as server is idle and there is a task in the queue

In the cases considered hereafter, we will restrict our attention to the class of stationary and deterministic policies that base their actions only on the current state of the system. Later we will also show that the found optimal policy is in fact optimal within a broader class that includes nonstationary and randomized policies as well when the service times are deterministic.

Let us first define an excursion from state x to be the event that starts with the acceptance of a task by the server at state x and ends with the first subsequent return to state x. The amount of time taken for such an excursion is defined to be $T(x, t, \tau)$ where t is the amount of time the server is busy. The significance of this will become more clear in subsequent sections. Also, given Equation (3.1), let us now examine the system behavior without the use of an optimal policy. As the server can only exist in two states, the server state varies in the following manner:

When the server is idle for a length of time t' ($b(t) = 0$):

$$x(t + t') = x(t)e^{-t'/\tau}$$

Alternatively, when the server is busy for a length of time t' ($b(t) = 1$):

$$x(t + t') = 1 - (1 - x(t))e^{-t'/\tau}$$

Hence from the two equations above, and assuming that the server accepts a single task when he is idle with an initial state of x_0 and is then busy for a time period of

t_1, the excursion time can then be calculated as follows:

$$T(x_0, t_1, \tau) = \tau \cdot \ln\left(\frac{1-(1-x_0)e^{-\frac{t_1}{\tau}}}{x_0}\right) + t_1 \tag{3.3}$$

From the server dynamics, we note that if we opt to use a greedy policy that allocates a task to the server every time it is idle (a policy which is very typical in the simplest of single-server queues) and assuming that the arrival rate of the tasks is high enough to always keep the server busy, then the server state will asymptotically converge to 1^-. This results in the server having a fixed service time of $\mathcal{S}(1^-)$ at steady state. Hence, without a more intelligent policy and in the long run as x converges to 1^-, $\lambda_{max} = 1/\mathcal{S}(1^-)$. However, since we assume that $\mathcal{S}(x)$ is convex and that $\mathcal{S}_{\min} := \min\{\mathcal{S}(x) \mid x \in (0,1)\}$, and $\mathcal{S}_{\max} := \max\{\mathcal{S}(0^+), \mathcal{S}(1^-)\}$, we note that in reality the server could perform each task with a shorter service time of $\mathcal{S}_{\min}$, which could imply a higher λ_{max}. Within the realm of stationary policies, this means that we would then be restricting our attention to policies u such that either $u(x) = 0$ for all $x > x_{threshold}$ for some value of $x_{threshold} \in (0,1)$, which ensures the existence of a maximally stabilizable arrival rate or such that $u(x) = 1$ for all $x \leq x_{lower}$, and remaining that way until $x > x_{upper}$ after which $u(x) = 0$, given that $x_{lower} < x_{upper}$. Note that in the case that $x_{lower} = x_{upper}$, the latter stationary policy would be similar to the former one. Hence, all this implies that λ_{max} is actually determined exactly by the excursion time, $T(x_0, t_1, \tau)$ when considering stationary policies and with deterministic service times. Note that we make the assumption here that t_1 is finite since the case where t_1 is infinite is already dealt with above using a greedy policy and provides the lower bound on the maximally stabilizable arrival rate.

Lemma 3.2.1. *For all $t_1 < \infty$, $x_0 \in (0,1)$, $\tau > 0$ and with deterministic arrivals, the maximally stabilizable arrival rate using a stationary threshold policy with threshold value x_0 is:*

$$\lambda_{max}(x_0, t_1, \tau) = \frac{n'(t_1)}{T(x_0, t_1, \tau)} \tag{3.4}$$

where $n'(t_1)$ is the number of tasks completed during the excursion. Note that for *stationary policies with deterministic service times, this value $n'(t_1)$ will be fixed.*

Proof. The proof of this lemma follows from that of a standard D/D/1 queue, where independent and identical tasks arrive at a deterministic rate of $\lambda > 0$ and the service time of each task is a constant $s > 0$. In order for the queue not to diverge, it is known that the maximum stabilizable arrival rate is $\frac{1}{s}$ such that a task is completed at the exact same moment that a new task arrives to the system. Hence, the queue length never changes and assuming that the queue was empty to begin with, the queue will remain empty always. In our formulation, however, the service times are state-dependent and the server state is a function of its utilization profile. Hence direct application of the D/D/1 result to obtain a similar stability condition is not apparent. However, the introduction of excursions simplifies this problem since the server's long-run behavior under a stationary threshold policy can be viewed as a series of similar excursions lined up one after another. Moreover, we also assume that the lengths of such excursions are finite since t_1 is finite and can be calculated deterministically as shown in Equation (4.1). Hence, as long as the queue length never grows after one excursion for a certain λ, we can deduce that the arrival rate is stabilizable. On the other hand, in order to obtain the maximally stabilizable arrival rate, it should also be apparent that the length of the excursion during which n' tasks were serviced should be exactly equal to the inter-arrival time between the i^{th} and $(i + n')^{th}$ arrivals. This is so that at the completion of one excursion, the first task of the next excursion arrives at that same moment, and hence preventing the server from unnecessary idleness.

$$\begin{aligned} n'(t) \cdot \frac{1}{\lambda(x,t,\tau)} &= T(x,t,\tau) \\ \lambda(x,t,\tau) &= \frac{n'(t)}{T(x,t,\tau)} \end{aligned}$$

Moreover, it should also be noted that during an excursion, depending on the interarrival times and the service times, the queue length could potentially grow. However, the queue length at the start and at the end of the excursion will definitely be the same subsequently. The queue length during an excursion will also never be infinite since the length of an excursion is assumed to be finite. □

Later, we will also show that such an analysis can be extended to the case of a stochastic server in order to derive a policy that gives a higher useful throughput compared to simply a greedy policy, though there exists no guarantees on its optimality. In addition, non-stationary policies such as n-task threshold policies, where $n \in \mathbb{N}$ will also be shown to be non-optimal. Finally note that in the case that $\mathcal{S}_{\min} = \mathcal{S}_{\max}$ and where we are only maximizing the throughput of the queue, $\lambda_{max} = 1/\mathcal{S}(1^-)$ is optimal and, in this case, $x_{threshold} = 1^-$.

3.3 Case 1: Convex Objective Function

Let us first begin by examining the case with deterministic state-dependent service times and error rates and where the server is assumed to have the following objective function:

$$\max_{(x,t)} f_u(t,\tau,x) := \max_{(x,t)} \left\{ c\overline{C_u(t,\tau,x)} + (1-c)\lambda_{max,u}(x,t,\tau) \right\} \tag{3.5}$$

Note that in the equation above, each term is treated to be a function of the state x since as mentioned in the section above, we will be primarily looking at stationary threshold policies, meaning that the policy taken at a certain server state x will always be the same. In addition, t is also assumed to be finite, and similar to Equation (4.1), t represents the amount of time the server is busy during one excursion. Note that since we are looking at stationary threshold policies with deterministic service times, every excursion from the same threshold value would be similar to each other, assuming that the server is constantly making the decision of whether or not to accept a task when it is idle. Hence, the server behavior in the long-run would simply be a series of similar excursions from the same threshold value and so for ease of analysis, we would subsequently just need to examine the behavior of the server during one excursion to be able to characterize its long-run behavior. Moreover, as can be seen from (3.5), in addition to determining the optimal threshold value, we would also need to determine the optimal excursion time from x.

The term $\overline{C_u(t,\tau,x)}$ can be treated as the average probability of getting a task correct in the excursion timeframe of $[t_0, t_0+T(x,t,\tau)]$ when the server is busy under policy u and with a threshold value of x and a busy time of t, assuming a task is accepted at time t_0 when the server state is x. Similarly $\lambda_{max}(\tau,u,x)$ is the maximally stabilizable arrival rate under the same circumstances with a threshold value of x. Given that we know that $\mathcal{S}(x)$ and $\mathcal{E}(x)$ are convex for $x \in (0,1)$, we can manipulate Equation (3.5):

$$\begin{aligned}\max_{(x,t)} f_u(t,\tau,x) &= \min_{(x,t)} \left\{ -[c\,(1-\overline{\mathcal{E}_u(t,\tau,x)}) + (1-c)\,\frac{n'(t)}{T(x,t,\tau)}] \right\} \\ &\approx \min_{(x,t)} \left[c\,\overline{\mathcal{E}_u(t,\tau,x)} + (1-c)\,\frac{T(x,t,\tau)}{n'(t)} + C \right]\end{aligned} \tag{3.6}$$

where C consists of constants that are inconsequential to the optimization problem and $n'(t)$ is defined to be the solution of the following minimization problem shown in Equation (3.7).

$$\begin{aligned} \min \quad & n' \\ \text{subject to} \quad & \sum_{i=0}^{n'-1} \mathcal{S}(x_i) \geq t \\ & x_{i+1} = 1-(1-x_i)e^{-\mathcal{S}(x_i)/\tau}. \end{aligned} \tag{3.7}$$

The average error rate $\overline{\mathcal{E}_u(t,\tau,x))}$ can then be found by Equation (3.8).

$$\begin{aligned} \overline{\mathcal{E}_u(t,\tau,x))} &:= \frac{\sum_{i=0}^{n'(t)-1} \mathcal{E}(x_i)}{n'(t)}. \\ x_{i+1} &= 1-(1-x_i)e^{-\mathcal{S}(x_i)/\tau} \end{aligned} \tag{3.8}$$

Given these definitions, the optimal choice of $x_{threshold}$ is then shown in Equation (3.9) and we are now primarily concerned with determining what is the optimal value of t, which resultantly helps us find the optimal value of $x_{threshold}$.

$$\begin{aligned} x_{threshold} &:= \arg\max_{x}\{f_u(t,\tau,x)\} \\ x_{threshold} &\in (0,1) \end{aligned} \tag{3.9}$$

From Equation (3.7), we note that the selection of the optimal value of t is akin to selecting the optimal value of $n'(t)$. This is because t is defined as the period of time that the server must be busy with tasks and since the service times of the server are deterministic, the value of n' is also uniquely determined. Hence, we will now focus primarily on determining the optimal number of tasks that should be done by the server before the server is given a break to complete the excursion from $x_{threshold}$.

3.3.1 Determination of optimal $n'(t)$

First, we show that the objective function that we seek to minimize in Equation (3.6) is convex for $x \in (0,1)$. Define:

$$F_1(x,t,\tau) := [\underbrace{c\,\overline{\mathcal{E}_u(x,t,\tau)}}_{\text{Term1}} + \underbrace{(1-c)\,\frac{T(x,t,\tau)}{n'(t)}}_{\text{Term2}} + \underbrace{C}_{Term3}] \tag{3.10}$$

From Equation (3.10), we know that since $c \geq 0$, Term 1 is a convex function with respect to x since the sum of convex functions is still convex. Also $n'(t)$ can be treated to be a constant for a given t. Since Term 3 is also approximated to be a constant, we can say that $F_1(x,t,\tau)$ is convex if Term 2 is shown to be convex.

Lemma 3.3.1. *For all $t > 0$, $x \in (0,1)$ and $\tau > 0$, given that $\tilde{T} : x \mapsto \tilde{T}(x) = T(x,t,\tau)$, then $\tilde{T}(x)$ is convex in x.*

Proof. The convexity of this function can be proven from its second derivative since $\tilde{T}$ is twice differentiable in the interval $x \in (0,1)$.

$$
\begin{aligned}
\tilde{T}(x) &= \tau \cdot \ln\left(\frac{1-(1-x_0)e^{-\frac{t}{\tau}}}{x_0}\right) + t \\
\frac{\delta \tilde{T}(x)}{\delta x_0} &= -\tau \frac{e^{\frac{t}{\tau}}-1}{x_0(e^{\frac{t}{\tau}}-1+x_0)} \\
\frac{\delta^2 \tilde{T}(x)}{\delta x_0^2} &= \tau \frac{e^{\frac{2t}{\tau}} - 2e^{\frac{t}{\tau}} + 2x_0 e^{\frac{t}{\tau}} + 1 - 2x_0}{(e^{\frac{t}{\tau}}-1+x_0)^2 x_0^2} \\
&= \tau \frac{(e^{\frac{t}{\tau}}-1)^2 + 2x_0(e^{\frac{t}{\tau}}-1)}{(e^{\frac{t}{\tau}}-1+x_0)^2 x_0^2}
\end{aligned}
$$

From $\frac{\delta^2\tilde{T}(x)}{\delta x_0^2}$, we see that for $t > 0$, $x \in (0,1)$ and $\tau > 0$, $\frac{\delta^2\tilde{T}(x)}{\delta x_0^2} > 0$, indicating that $\tilde{T}(x)$ is strictly convex. □

Given that the objective function $F_1(x,t,\tau)$ is convex, we can then say that $F_{1,min} := \min\{F_1(x,t,\tau) \mid x \in (0,1)\}$, and $F_{1,max} := \max\{F_1(0^+,t,\tau), F_1(1^-,t,\tau)\}$. Next, let us re-define the excursion time to be a function of $n'(t)$, $T_1(x,n,\tau)$, instead of t, $T(x,t,\tau)$, where $n := n'(t)$ and is determined using Equation (3.7).

$$
\begin{aligned}
T_1(x_0,n,\tau) &= \tau \cdot \ln\left(\frac{1-(1-x_0)e^{-\frac{\sum_{i=0}^{n-1}\mathcal{S}(x_i)}{\tau}}}{x_0}\right) + \sum_{i=0}^{n-1}\mathcal{S}(x_i) \\
x_{i+1} &= 1-(1-x_i)e^{-\mathcal{S}(x_i)/\tau}
\end{aligned}
\tag{3.11}
$$

Hence, our objective function can now be written as:

$$
\begin{aligned}
\min_{(x,t)}\{F_1(x,t,\tau)\} : &= \min_{(x,n)}\left\{c\,\overline{\mathcal{E}_u(x,n,\tau)} + d\frac{T(x,n,\tau)}{n} + C\right\} \\
&= \min_{(x,n)}\left\{c\,\overline{\mathcal{E}_u(x,n,\tau)} + d\frac{T(x,n,\tau)}{n}\right\}
\end{aligned}
\tag{3.12}
$$

Next, we will examine two interesting properties that will help us prove the optimal policy. The first property is that given the server state dynamics in Equation (3.1), the cost to complete n tasks consecutively in one excursion will always be greater or equal to the cost to complete the same number of tasks when 1 task is done per excursion, where $n \in \mathbb{N}$. This property will be useful in proving the optimal policy. Next, the second property is that given a certain excursion, a time-shift in the start of the busy time of the excursion will not affect the cost of the excursion. This property in turn helps to prove the optimality of the 1-task threshold policy.

Lemma 3.3.2. *For $x_0 \in (0,1)$ such that $x_0 = \arg\min_x\{F_1(x,1,\tau)\}$, $n \in \mathbb{N}$ and $\tau > 0$, $F_1(x_0,n,\tau) \geq F_1(x_0,1,\tau)$, assuming that the former policy $u \in \mathcal{U}$ assigns a task to the server right away as long as the number of tasks assigned to the server so far is $\leq n$. Note that we also assume the arrival rate to be λ_{max} and hence the arrival rate is high enough such that there always is a task in the queue to assign to the server when desired.*

Proof. First, we define $x_0 = \arg\min_x\{F_1(x,1,\tau)\}$. Hence, this implies that any other choice of x as the starting threshold value would result in a greater or equal cost. Next, since we know that $F_1(x,n,\tau)$ is convex for any given $n \in \mathbb{N}$, we know that this minimum exists and that $x_0 \in (0,1)$. Hence, we can then prove the following lemma by mathematical induction.

We begin by comparing the non-trivial case of $n = 2$, that is we want to show that $F_1(x_0,2,\tau) \geq F_1(x_0,1,\tau)$. Next, let us define the notation x_1 and x_2 to represent the server state at the conclusion of the first and second tasks respectively. Note that in the case of n tasks, x_n would represent the server state at the conclusion of the n^{th} task. Hence, when the server is busy, its state goes from x_0 to x_1 and then to x_2, where $x_0 < x_1 < x_2$.

$$\begin{aligned}
F_1(x_0,2,\tau) &= c\overline{\mathcal{E}(x_0,2,\tau)} + d\frac{T(x_0,2,\tau)}{2} \\
&= c\frac{\sum_{i=0}^{1}\mathcal{E}(x_i,1,\tau)}{2} + d\frac{\tau\cdot\ln(x_2/x_0) + \sum_{i=0}^{1}\mathcal{S}(x_i)}{2} \\
&= c\frac{\sum_{i=0}^{1}\mathcal{E}(x_i,1,\tau)}{2} + d\frac{\tau\cdot(\ln(x_2/x_1) + \ln(x_1/x_0)) + \sum_{i=0}^{1}\mathcal{S}(x_i)}{2} \\
&= \frac{1}{2}(F_1(x_0,1,\tau) + F_1(x_1,1,\tau)) \\
&\geq F_1(x_0,1,\tau)
\end{aligned}$$

The inequality at the end holds true from the definition of $x_0 = \arg\min_x\{F_1(x,1,\tau)\}$, which implies that $F_1(x_1,1,\tau) \geq F_1(x_0,1,\tau)$. Hence, the lemma holds true for $n = 2$. Now consider $n = k+1$ assuming that the lemma holds true for $n = k$. Now, we want to prove that $F_1(x_0, k+1, \tau) \geq F_1(x_0, 1, \tau)$, given that $F_1(x_0, k, \tau) \geq F_1(x_0, 1, \tau)$.

$$\begin{aligned}
F_1(x_0, k+1, \tau) &= c\,\overline{\mathcal{E}(x_0, k+1, \tau)} + d\,\frac{T(x_0, k+1, \tau)}{k+1} \\
&= c\,\frac{\sum_{i=0}^{k} \mathcal{E}(x_i, 1, \tau)}{k+1} + d\,\frac{\tau \cdot \ln(x_{k+1}/x_0) + \sum_{i=0}^{k} \mathcal{S}(x_i)}{k+1} \\
&= \frac{1}{k+1}(c\sum_{i=0}^{k-1} \mathcal{E}(x_i, 1, \tau) + d(\sum_{i=0}^{k-1} \mathcal{S}(x_i) + \tau \ln(\frac{x_k}{x_0})) + \\
&c\mathcal{E}(x_k, 1, \tau) + d(\mathcal{S}(x_k) + \tau \ln(\frac{x_{k+1}}{x_k}))) \\
&= \frac{1}{k+1}(k \cdot F_1(x_0, k, \tau) + F_1(x_k, 1, \tau)) \\
&\geq F_1(x_0, 1, \tau)
\end{aligned}$$

Similarly, the last inequality holds true from the definition of x_0 which implies that $F_1(x_{k+1}, 1, \tau) \geq F_1(x_0, 1, \tau)$ and also since it is assumed that $F_1(x_0, k, \tau) \geq F_1(x_0, 1, \tau)$. Hence, this concludes the proof of this lemma, which indicates that given the server dynamics, the optimal policy should be a 1-task threshold policy, since attempting to complete 2 tasks within 1 excursion will never give a better cost than completing the same 2 tasks using 2 excursions, where 1 task is completed per excursion. □

The next lemma proposes that given a certain excursion, a time-shift in the start of the busy time of the excursion will not affect the cost of the excursion. It should be noted however that the server cannot be interrupted during service of a task. Hence, this time-shift in reality only pertains to the policy, $\tilde{u}$, which waits for some initial time until the server state reaches state x_0^- from state x_0 before assigning the task to the server.

Lemma 3.3.3. *For any* $x \in (0, 1)$, $F_{1,u}(x, 1, \tau) = F_{1,\tilde{u}}(x', 1, \tau)$ *iff* $x' > x$ *and* $x'^- = x$

Proof. Let x_1 represent the server state at the completion of the task.

$$\begin{aligned}
F_{1,\tilde{u}}(x', 1, \tau) &= \tau \ln(\frac{x'}{x}) + \mathcal{S}(x) + \tau \ln(\frac{x_1}{x'}) \\
&= \mathcal{S}(x) + \tau \ln(\frac{x_1}{x}) \\
&= F_{1,u}(x, 1, \tau)
\end{aligned}$$

□

From the above two lemmas, we can infer a third lemma. First as before, we analyze only one excursion and assume that a task is assigned to the server at the start of the excursion, which is a reasonable assumption given the lemma above. This is because if the policy $\tilde{u}$ lets the server idle initially until it reaches a certain state, x^-, then one can instead consider the same problem with initial state x^-. Next, the lemma states that the total cost of an excursion from state x during which n tasks are done under policy $\hat{u}$ with each task being assigned at state x_i is equal to the cost of n excursions, where the i^{th} task is assigned during the i^{th} excursion with starting server state x_i.

Lemma 3.3.4. *For any policy $\hat{u}$ that assigns n tasks during an excursion, $F_{1,\hat{u}}(x_0, n, \tau) = \sum_{i=0}^{n-1} F_{1,u}(x_i, 1, \tau)$, given that x_i is the server state when the $(i+1)^{th}$ task is assigned. Note that policy u is similar to that in Lemma 3.3.2.*

Proof. To begin, we note that there do not exist any assumptions on the type of policy that $\hat{u}$ should be, apart from the fact that it assigns n tasks to the server during an excursion. From earlier, an excursion is defined to be the event from state x that starts with the acceptance of a task by the server at state x and ends with the first subsequent return to state x. Hence, a general form of the policy $\hat{u}$ can be defined to be a concatenation of a task assignment followed by a fixed period of idleness for the server. Note that in accordance with its definition, the period of idleness during an excursion should never be long enough such that the server state returns to x_0 before the n tasks are assigned. In addition, the periods of idleness can also be taken to be 0 between tasks. In fact, the policy u can be seen to be a variant of $\hat{u}$ where each fixed period of idleness is taken to be 0 until all n tasks are assigned, after which the server is given a long period of idleness to return to state x_0.

From the general form of the policy $\hat{u}$, let us first assume that the periods of idleness are never 0. First, we note that when $n = 1$, the equality holds true trivially. Next, consider $n = 2$. Let us denote the server state at the assignment of the i^{th} task to be x_i while the server state at the completion of the i^{th} task is x_i'. Hence, for the case of $n = 2$, under policy $\hat{u}$, the server state will go from x_0 up to x_0' down to x_1 up to x_1' and finally back down to x_0 during the excursion.

$$\begin{aligned}
F_{1,\hat{u}}(x_0, 2, \tau) &= \mathcal{S}(x_0) + \tau \ln(\frac{x_0'}{x_1}) + \mathcal{S}(x_1) + \tau \ln(\frac{x_1'}{x_0}) \\
&= \mathcal{S}(x_0) + \mathcal{S}(x_1) + \tau \ln(\frac{x_0'}{x_0}) + \tau \ln(\frac{x_1'}{x_1}) \\
&= \sum_{i=0}^{1} F_{1,u}(x_i, 1, \tau)
\end{aligned}$$

Given that the lemma holds true for $n = 2$, now consider the case $n = k+1$ assuming that the lemma holds true for $n = k$, that is we want to prove that $F_{1,\hat{u}}(x_0, k+1, \tau) = \sum_{i=0}^{k} F_{1,u}(x_i, 1, \tau)$ given that $F_{1,\hat{u}}(x_0, k, \tau) = \sum_{i=0}^{k-1} F_{1,u}(x_i, 1, \tau)$.

$$\begin{aligned}
F_{1,\hat{u}}(x_0, k+1, \tau) &= \sum_{i=0}^{k-1} (\mathcal{S}(x_i) + \tau \ln \frac{x_i'}{x_{i+1}}) + \mathcal{S}(x_k) + \tau \ln(\frac{x_k'}{x_0}) \\
&= \sum_{i=0}^{k} (\mathcal{S}(x_i) + \tau \ln(\frac{x_i'}{x_i})) \\
&= \sum_{i=0}^{k} F_{1,u}(x_i, 1, \tau)
\end{aligned}$$

Hence, by mathematical induction, this lemma holds true and similar to Lemma 3.3.2, implies that the optimal policy should be of the form of a 1-task threshold policy since the cost will only be minimum in this case if $x_{th} = \arg\min_x \{F_1(x, 1, \tau)\}$ and $n = 1$. The choice of any other n will result in the summation of a cost $F_1(x_{nth}, 1, \tau)$ with $F_1(x_{th}, 1, \tau)$ where $x_{nth} \neq x_{th}$, which by definition would result in a greater overall cost than if we were to adopt a 1-task threshold policy. □

3.3.2 Optimal Policy

Given the lemmas defined in the previous section, we propose a simple 1-task threshold policy that can be stated as follows:

$$u_{\mathrm{TP}}(t) = \begin{cases} \text{ON} & \text{if } x(t) \leq x_{\mathrm{th}}(\tau), \\ \text{OFF} & \text{otherwise,} \end{cases}$$

where $x_{\mathrm{th}}(\tau)$ is as defined in Equation (3.9). We subsequently show that this policy is the optimal policy given our objective function in Equation (3.12) and will further term this policy to be a 1-task threshold policy since based on the server dynamics and with an arrival rate of λ_{max}, the assignment of a task to the server at x_{th} can be treated as an excursion for a single task from x_{th}. The maximally stabilizing λ is then found from Equation (3.4).

$$\lambda_{max}(x_{th}, \mathcal{S}(x_{th}), \tau) = \frac{1}{T(x_{th}, \mathcal{S}(x_{th}), \tau)} \tag{3.13}$$

Theorem 3.3.5. *For $\tau > 0$, $x_0 \in (0,1)$, $x_{th} \in (0,1)$, $q_0 \in \mathbb{N}$ and $\lambda \leq \lambda_{max}(x_{th}, \mathcal{S}(x_{th}), \tau)$, $\limsup_{t \to +\infty} q_u(t, \tau, \lambda, x_0, q_0) < +\infty$.*

Proof. First let us define x_i and t_i to be the server state and time instants respectively at the beginning of service of the i_{th} task. Also, let $q(t)$ represent the queue length at time t. Assume that the server state initially is $x_0 \in (0,1)$ and that the initial queue length q_0 is finite. Hence, without loss of generality, let us assume that $x_0 > x_{th}$. Based on the policy defined earlier, no task is assigned to the server until the server state is x_{th}. Hence, the queue length when the first task is assigned to the server can then be defined as $q(t_1) = max\{0, q_0 - 1, q_0 - 1 + \lfloor \lambda\tau \ln(\frac{x_0}{x_{th}}) \rfloor\}$. Note that the second entry in the maximization corresponds to no tasks arriving during the idle period that the server state decreases from x_0 to x_{th}, while the third entry corresponds to the case where the idle time is long enough for tasks to arrive, which then end up piling up in the existing queue. Now, we will prove that $q(t_i) \leq q(t_1) + \lceil(-\tau \ln(1 - x_{th}) + \mathcal{S}_{max})\lambda\rceil + \lceil -\lambda\tau \ln(x_{th})\rceil$ for all i by considering the following two cases:

- **State 1**: $x_1 = x_{th}$. This case occurs when a task arrives during the period that the server is idle or if the queue was not empty to begin with. If $\lambda = \lambda_{max}$, then the inter-arrival time between tasks is the same as the excursion time under a 1-task threshold policy and hence $q(t_i) \equiv q(t_1)\ \forall i$. If $\lambda < \lambda_{max}$, then the inter-arrival time between tasks is more than the excursion time and hence there exists an $i' \geq 1$ such that $q(t_i) < q(t_{i-1})\ \forall i \leq i'$ and $q(t_{i'} + T(x_{th}, \mathcal{S}(x_{th}), \tau)) = 0$ and hence $x_{i'+1} < x_{th}$ due to the time that the server is allowed to be idle after the queue is emptied. Thereafter, we will consider this case to be similar to the next case by resetting $x_{i'+1}$ and $t_{i'+1}$ as x_1 and t_1 respectively. $q(t_1)$ in the next case will hence begin at 0.

- **State 2**: $x_1 < x_{th}$. This case occurs when a task arrives to an empty queue when the server state is $< x_{th}$ or in the event that $x_0 < x_{th}$. This is because the server will never be idle when the queue length is non-zero. Hence, here we seek to find an upper bound on the maximum number of outstanding tasks possible when the server state reaches x_{th}. First, we note that the maximum amount of continuous service time required for the server state to cross x_{th} starting from any $x_1 < x_{th}$ is upper bounded by $-\tau \ln(1 - x_{th}) + \mathcal{S}_{max}$ where $x_0 = 1^-$. It could then be followed by an idle time which is upper bound by $-\tau \ln(x_{th})$, at the end of which the server state is x_{th}. Hence, during this total period, the maximum number of outstanding tasks that accumulates in the queue by the time the server state reaches x_{th} is upper bounded by $q(t_i) \leq q(t_1) + \lceil(-\tau \ln(1 - x_{th}) + \mathcal{S}_{max})\lambda\rceil + \lceil -\lambda\tau \ln(x_{th})\rceil$. Thereafter, we will consider this case to be similar to the earlier case with $x_1 = x_{th}$ and n_1 to be the number of outstanding tasks when the server state reaches x_{th}.

Hence, in summary, when the system is in State 1 and if $\lambda = \lambda_{max}$, then the queue length stays constant. Otherwise, the queue length will monotonically decrease to zero at which point it enters State 2. On the other hand, if the system is in State 2, it stays in it forever or eventually enters State 1 with bounded queue length. Hence, this proves the theorem that the queue length will always be finite. $\square$

Theorem 3.3.6. *Define $u \in \mathcal{U}$ to be any policy that achieves the 1-task threshold policy as defined above. For $\tau > 0$ and $x_{th} = \arg\min_x\{F_1(x, 1, \tau)\}$, $F_{1,u}(x_{th}, 1, \tau) \leq F_{1,u'}(x, n, \tau)$ where $x \in (0, 1)$, $n \in \mathbb{N}$ and $u' \in \mathcal{U}$.*

Proof. First, we prove that the policy u is optimal within the class of all stationary policies. This encompasses policies such as a single x-threshold policy or dual switching threshold policies. Furthermore, the assignment of tasks within each excursion can also vary from policies that assign all available tasks one after another immediately to policies that give the server a certain period of idle time upon each completion of a task. From Lemma 3.3.3, we can state without any loss of generality that every policy begins with the immediate assignment of a task to the server since any policy that allows the server to have a period of idleness until state x_0^- can be redefined as the same policy but with a starting state of x_0^-. Hence, given deterministic service times of the server, the only difference between various stationary policies would then be the number of tasks done within one excursion and the amount of idle time between tasks. From Lemma 3.3.2 and Lemma 3.3.4, we note that $F_{1,u}(x_{th}, 1, \tau) \leq F_{1,u}(x_{th}, n, \tau)$ and $F_{1,u}(x_{th}, 1, \tau) \leq \sum_{i=0}^{n-1} F_{1,u}(x_i, 1, \tau) = F_{1,\hat{u}}(x_{th}, n, \tau)$ where $x_0 = x_{th}$ and $n \in \mathbb{N}$ respectively, hence proving the optimality of u among the class of stationary policies.

The same reasoning can be extended to the class of all non-stationary policies. A non-stationary policy, $\breve{u}$ is a policy that assigns a task to the server at $x_{th}(t)$. Let us define x_i and t_i to be the server state and time at which the server is assigned the i^{th} task. Next, using the same mathematical tricks as Lemma 3.3.3 and Lemma 3.3.4, we note that during a timeframe of $[t_i, t_j]$, if $x_i = x_j$, then $F_{1,\breve{u}}(x_i, n_{ij}, \tau) = \sum_{i'=i}^{j-1} F_{1,u}(x_{i'}, 1, \tau)$. This event will occur infinitely often under $\breve{u}$ and hence $F_{i,u}(x_{th}, 1, \tau) \leq \sum_{i'=i}^{j-1} F_{1,u}(x_{i'}, 1, \tau) = F_{1,\breve{u}}(x_i, n_{ij}, \tau)$.

Finally, a greedy policy will also be shown to be suboptimal. A greedy policy is defined to be a policy that assigns a task to the server as long as there exists a task in the queue. To show this, let us assume an arrival rate λ_{max} according to Equation 3.13. Next, we consider the following 3 cases of the initial server state x_0:

- **Case 1:** $x_0 < x_{th}$: Let us define $x(\infty)$ to be the server state at $t = \infty$. In this case, $x(\infty) = 1^-$ because the queue length will never be empty upon arrival of the second task, based on the definition of λ_{max}. Hence the server will always be busy and $x \rightarrow 1^-$.

- **Case 2:** $x_0 = x_{th}$: Assuming $q_0 = 0$, $x(\infty) = x_{th}$ and this case is exactly similar to the 1-task threshold policy proposed above.

- **Case 3:** $x_0 > x_{th}$ Similar to Case 1, $x(\infty) = 1^-$ since the queue length will never be empty upon arrival of the second task.

Hence, for Cases 1 and 3, we note that the cost of operating at $x_{th} = 1^-$ under a greedy policy is equivalent to $F_{1,u}(1^-, 1, \tau)$. From observation, we can then say that a greedy policy is suboptimal if $1^- \neq \arg\min_x\{F_1(x, 1, \tau)\}$. Thus this concludes the proof of the optimality of the 1-task threshold policy among all possible policies in $\mathcal{U}$. $\square$

3.3.3 Extension 1: Switching Costs

Given the optimal policy stated in Section 3.3.2, it is interesting to note two key assumptions made. The first is that the server state is being continuously monitored and the second is that there exist neither any costs in keeping a task waiting in the queue, nor any costs in continually switching the server between periods of busyness and idleness. This section of this thesis hence explores variants of the optimal policy that can be employed in the event that the latter assumption does not hold true and that such costs do need to be accounted for.

As shown from the literature review in Chapter 2, the goal of queueing systems is to find methods to reduce delays for customers and to keep long-run costs down. For example, in a jobshop setting, costs may be incurred if a product is unable to be completed on time due to a bottleneck that happened somewhere along the process chain. Such costs could be the costs of not being able to meet a scheduled deadline or loss of investor confidence. Moreover, when dealing with machinery or even when the server is a human, costs may be incurred for continually switching on and off the server. This could be due to the additional energy required to 'boot-up' the machine from a dormant state compared to if the machine has just been allowed to stay 'active' even when it was not servicing a job. In the case of human servers, a common explanation would be that the interruption of the server's 'train of thought' everytime he or she is given a short break after service of one task is overall more harmful to the server's productivity compared to the benefits of implementing the optimal policy derived in Section 3.3.2.

The problem with switching costs can be formulated as follows: Associate a cost of c_s for each time the server is switched on or off under the optimal policy derived in Section 3.3.2, where the server is defined to be switched off everytime it is not busy servicing a task. Each task completed is given a reward of y units, though only tasks that are completed correctly are rewarded. In this case, the policy derived earlier would hence not necessarily be optimal and a dual switching threshold policy as mentioned in Section 3.2 might be a better choice. Let G_{opt} represent the reward of one excursion, associated with the policy derived in Section 3.3.2 and let G_{dual} represent the reward of one excursion, during which n tasks are serviced using a dual switching threshold policy. For now, assume that the arrival rates in both cases are such that the queue is always kept stable. It is apparent that such an assumption implies different arrival rates for both policies but this will be dealt with subsequently. The reward functions of such a problem would hence look like Equation 3.14 and Equation 3.15.

$$G_{opt} = [1 - \mathcal{E}(x_{th})] \cdot y - 2 \cdot c_s \tag{3.14}$$

$$G_{dual} = \sum_{i=0}^{n-1} [1 - \mathcal{E}(x_i)] \cdot y - 2 \cdot c_s \tag{3.15}$$

Hence, it is apparent that the optimal policy for a system with switching costs is a single-threshold policy if $n \cdot G_{opt} \geq G_{dual}$ and is a dual switching threshold policy otherwise. We note that these two policies can be argued to be optimal within the realm of all stationary and non-stationary policies using the same arguments as before.

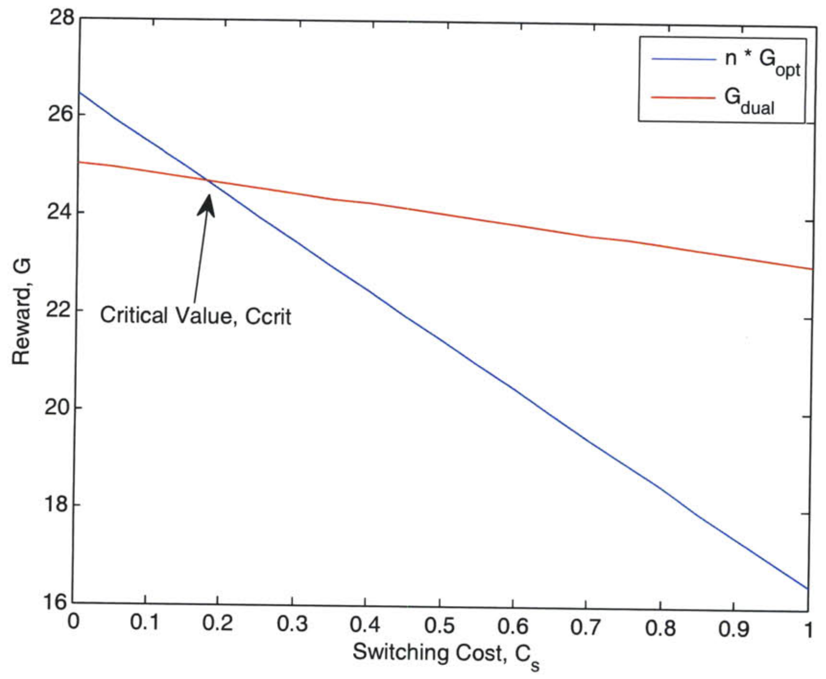


Figure 3-1: Comparison of Reward Functions of $n \cdot G_{opt}$ and G_{dual} for $n = 5$, $\tau = 300$, $y = 10$, $\mathcal{E}(x) = 1.9x^2 - 1.9x + 0.9$ and based on the service time function from Figure 1-3

The interpretation of such a policy is that a single-threshold policy is optimal if the benefit of completing a task quickly and correctly far outweighs the cost of frequently switching the server on and off. On the other hand, if switching costs are high, for example if a human server prefers to tackle a series of tasks at one go instead of being given time to rest after every task, then a policy that allows the server to service n tasks consecutively before allocating a period of rest is optimal. Figure 3-1 shows an illustration of such a phenomenon, whereby the single-threshold policy is optimal up until the critical value, after which the switching costs are so high that it is cheaper to implement a dual switching threshold policy.

In addition, the claim that the arrival rates in both cases are always such that the queue is kept stable is valid as long as the queue length at the start and end of an excursion are kept the same. Moreover, there must also be sufficient number of tasks to be allocated in the queue for the dual switching threshold policy to work. Hence, we notice that while Lemma 3.4 holds true for the single threshold policy, using the same formulae to derive the arrival rate for the dual switching threshold policy does not actually hold true, since it can be observed that the server would have no tasks to service for a period of time upon completion of the first task. Hence, for the latter case to work, let us assume a batch arrival system where n tasks arrive at an arrival rate of λ_{max} as defined by the single threshold policy. The difference in arrival processes, however, does not affect the selection of the optimal policy based on our definition of the reward function since the server is rewarded for each correctly done task.

3.3.4 Extension 2: Switching and Holding Costs

Given the assumption of a batch arrival system for the dual switching threshold policy case, it is now imperative to note that there would almost always be tasks waiting to be serviced in the queue, although in the long run, the queue length will never diverge to infinity. However, the delays associated with having to wait for service in the queue may be of concern and there may be costs associated with such delays. Let us define c_D to be the cost associated with a delay of 1 unit time for each task waiting in the queue, otherwise known in literature to be the holding cost. The reward function for the dual switching policy can then be modified as shown in Equation 3.16.

$$G_{dual} = \sum_{i=0}^{n-1}[1 - \mathcal{E}(x_i)] \cdot y - 2 \cdot c_s - c_D \cdot \sum_{i=1}^{n}(n-i)\mathcal{S}(x_{i-1}) \qquad (3.16)$$

Hence, the optimal policy for a system with switching and holding costs would once again be a single-threshold policy if $n \cdot G_{opt} \geq G_{dual}$ and would be a dual switching threshold policy otherwise. Figure 3-2 illustrates the cost variation for the same setup as that in Figure 3-1 though this time for varying n. The switching cost is now taken to be a constant of 0.5 and the holding cost is 0.005 per second per task. From the figure, we notice that for some n, the reward of implementing a dual switching threshold policy is higher than that of the single threshold policy, even when holding costs are considered in the reward function. However, as expected, the reward of implementing a dual switching threshold policy with holding costs is always less than that of one without and that the reward of the former always diminishes

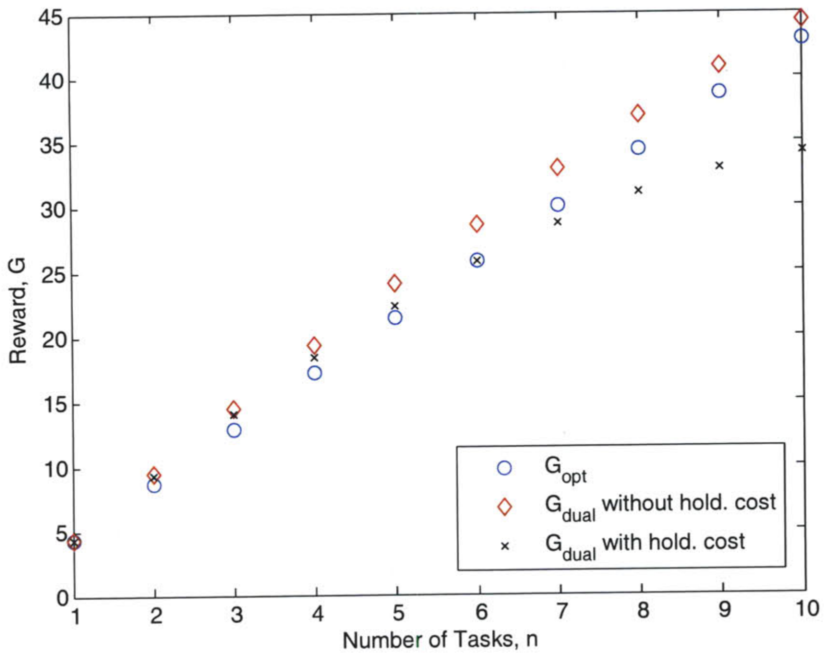


Figure 3-2: Comparison of Reward Functions of $n \cdot G_{opt}$ and G_{dual} for varying n with $c_s = 0.5$ and $c_D = 0.005/sec$

at an increasing rate as n increases. Hence, this implies that in the case that delays should be prevented and if the optimal policy is indeed the dual switching threshold policy, then the number of tasks done per excursion, n, would always be less than or equal to the number of tasks done per excursion if holding costs are not considered.

3.4 Case 2: General Objective Function

In this section, we relook at the assumption that the service time and error functions have to be convex graphs and instead propose that the same arguments mentioned throughput the chapter thus far can also be extended to cases where the objective function is not convex. Though much literature as mentioned earlier in the chapter validates the use of convex functions to represent the service time and error functions, there may be instances where the assumption may not hold true. Hence, we seek to show that the optimal policy still remains optimal, even for a general objective function.

First, let us begin by characterizing what a general objective function means.

We define a general objective function to be any function that is neither convex nor concave, meaning that multiple local minimum and maximum may exist. Assuming that such an objective function is observable and characterizable in reality, then the optimal policy can be stated as follows:

$$u_{\mathrm{TP}}(t) = \begin{cases} \text{ON} & \text{if } x(t) \leq x_{\mathrm{th}}(\tau), \\ \text{OFF} & \text{otherwise,} \end{cases}$$

$$x_{threshold} := \arg\min_{x}\{F_u(t, \tau, x)\}$$

$$x_{threshold} \in (0, 1)$$

We note that the above policy is exactly the same as that defined for a convex objective function. This is because Lemma 3.3.2 still holds true regardless of the convexity of the objective function, as long as $F(x_{k+1}, 1, \tau) \geq F(x_{th}, 1, \tau)$, which is true by definition. Next, Lemma 3.3.4 also holds true since the 'memoryless property' of the server is inherit in its server dynamics and not in the objective function. Hence, these two lemmas, in combination with Lemma 3.3.3, together verify the optimality of the above policy within the class of stationary and non-stationary policies. We now investigate the behavior of a state-dependent queue with a general objective function under a greedy policy.

As mentioned in Section 3.2, a greedy policy is one that allocates a task to the server every time it is idle , assuming that the arrival rate of tasks is high enough to always keep the server busy. Hence, the server state will converge to 1 in the long run, resulting in the server having a fixed cost objective function of $F_{greedy}(t, \tau, \lambda, x_0, 1^-)$. Hence, in the case that $x_{threshold} = 1^-$, then a greedy policy will perform similar to the single threshold policy. Otherwise, a single threshold policy is optimal since $F_u(t, \tau, x_{threshold}) \leq F_{greedy}(t, \tau, \lambda, x_0, 1^-)$ by definition. This concludes the proof that the same optimal policy derived for a convex objective function can be extended to cases where the objective function is of a general shape.

Chapter 4

Policies for Servers with Stochastic Performance

In Chapter 3, we assumed that the server's service rate at any state x was deterministic. However, this is typically invalid in reality, particularly if the server is a human since no one can attest to behaving in the exact same way and taking the exact same amount of time on a task all the time. Instead, a more apt model would be one where the server's service rate is assumed to be stochastic with a certain mean and variance. The state-dependence of such a queueing system is then modeled into the means of the service rate, which are now assumed to vary according to the Yerkes-Dodson law just as proposed in [19]. Moreover, if one refers back to Figure 1-3, it should be noted that the authors did notice that the distribution of decision times was well-modeled within each bin by a log-normal distribution as shown in Figure 4-1.

A lognormal distribution is popular for use in service time modeling due to its existence within the quadrant of positive $x-$ and $y-$ values, meaning negative service times will be not modeled, unlike if we were to use a normal distribution. An illustration of how the service times are envisioned to be distributed in reality is shown in Figure 4-2. This chapter will hence attempt to extend the work done in Chapter 3 to a case with a stochastic server, particularly examining the behavior of such a server using the optimal policy derived for a deterministic server in Chapter 3. Bounds on the maximally stabilizable arrival rate for a stochastic server will then be proposed.

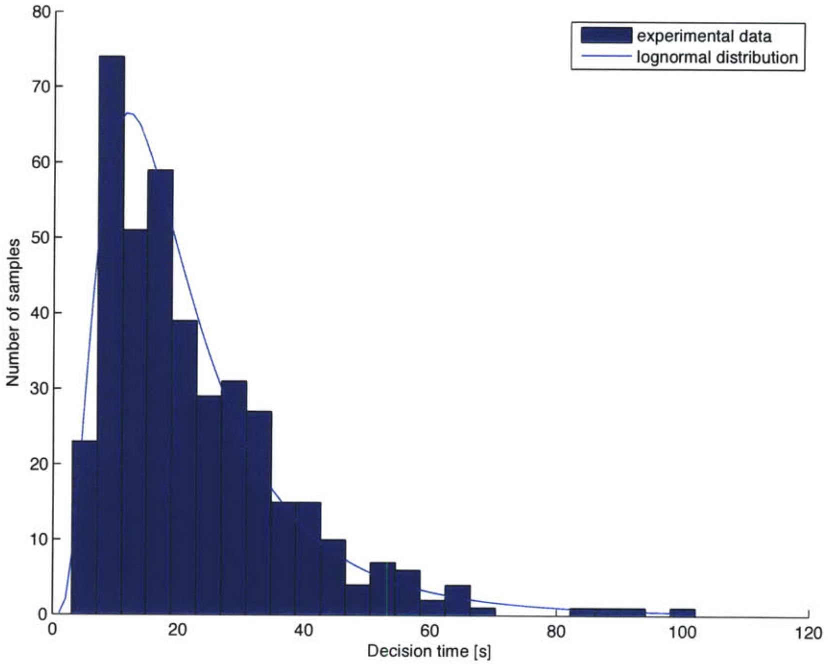


Figure 4-1: Distribution of decision time within each 5% utilization bin well-modeled by a lognormal distribution [19]

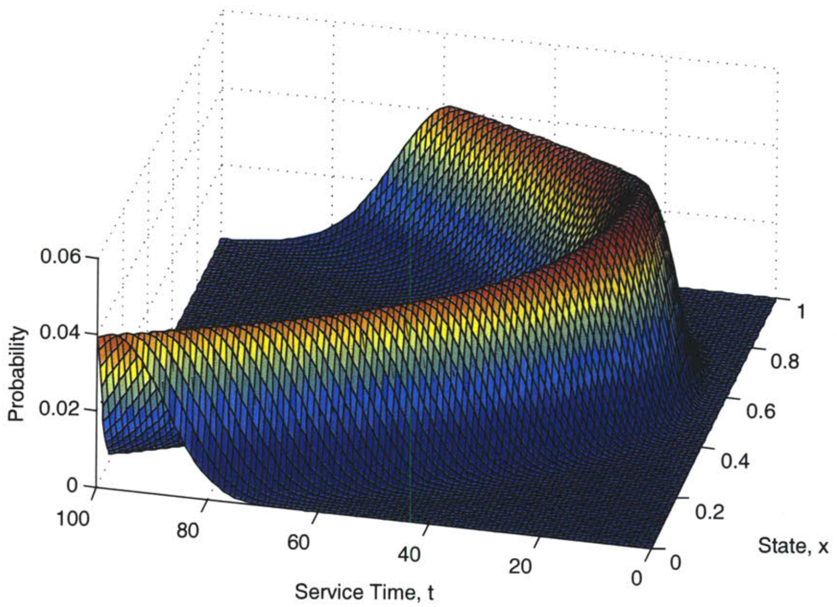


Figure 4-2: Illustration of service time distributions with lognormal mean, $m = 229x^2 - 267x + 99$ and lognormal standard deviation, $v = 10$

4.1 Extension of Single-Task Threshold Policy to Stochastic Server

In this section, we examine the outcome when the single-task threshold policy that was developed for a deterministic server is implemented for a stochastic server. Key differences of having a stochastic server are that firstly, the actual service times for each task is unknown to the controller until the task has been completed and secondly, that optimal control would require some form of closed-loop feedback, unlike the open-loop setup for a deterministic server, since the server's behavior is probabilistic.

In this chapter however, we modify our objective function to be one where we want to maximize the total throughput through the system, hence neglecting the error rates. This modification is done primarily to simplify our analysis subsequently when deriving bounds for the maximally stabilizable arrival rate for the stochastic server. Also, we make the second assumption that the variation of the service times with the state is convex for simplicity. Based on these assumptions, we can then envision three possible scenarios that could occur with a stochastic server and can provide some analysis.

Let us assume that tasks are arriving at the rate λ_{max} as defined in Equation 3.4 with an objective function of purely wanting to maximize the throughput and an associated threshold value of x_{th}. The desired service time for each task is hence $\mathcal{S}(x_{th})$ though in reality, the actual service time is $s_{stoc}(x_{th})$, where $\mathcal{S}_{stoc}(x_{th}) \sim \text{Log-}\mathcal{N}(\mu, \sigma^2)$. Finally, let x_i represent the server state after $\frac{i}{\lambda max}$ units of time from the time a task is allocated to the server at state x_{th}. In the ideal case, $x_i = x_{th}$ always, which is the case for a deterministic server, allowing for predictable server behavior that can hence be controlled.

The three possible scenarios when dealing with a stochastic server are as follows:

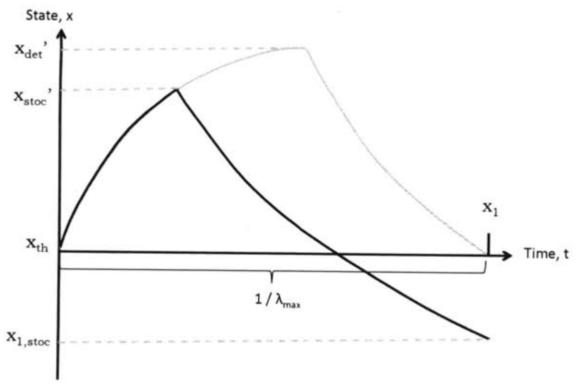
1. $\boldsymbol{s}_{stoc}(x_{th}) < \mathcal{S}(x_{th})$: In this case, $x_1 < x_{th}$ if the queue is empty since the total excursion time required for the server to have returned to x_{th} is less than each interarrival interval. This should be apparent given Equation 4.1 and Lemma 3.3.4 and is illustrated in Figure 4-3(a). Assuming the queue to have

been empty to begin with, the server would then remain idle until a new task entered the system.

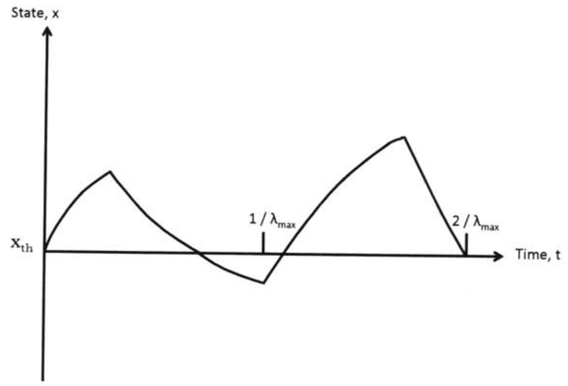
Under the single-task threshold policy, however, we term this case to be stabilizing but uncontrollable. This is because assuming that the queue is empty and as can be seen from Figure 4-3(b) , should the server continually be servicing the tasks faster than expected, then there exists a task i, $i \geq 1$ that will be serviced starting from a state $x_i < x_{th}$, such that $\overline{\mathcal{S}_{stoc}(x_i)} > \overline{\mathcal{S}_{stoc}(x_{th})}$, where our notation is that $\overline{\mathcal{S}_{stoc}(x)}$ is the mean of the random variable $\mathcal{S}_{stoc}(x)$. Note that even though $x_{th} = \arg\min_x\{T(x,t,\tau)\}$ (recall that the objective function we are minimizing is the excursion time since we are purely maximizing the throughput), $\overline{\mathcal{S}_{stoc}(x_1)}$ need not necessarily be greater than $\overline{\mathcal{S}_{stoc}(x_{th})}$. Hence, as Figure 4-3(b) depicts, this case is considered stabilizing since a server that is constantly under-servicing his tasks will continually have excessive amount of idle time, resulting in a corresponding decrease in the server state below x_{th}. When the server state is x_i, $\overline{\mathcal{S}_{stoc}(x_i)} > \overline{\mathcal{S}_{stoc}(x_{th})}$, indicating that either the server state $x_{i+1} = x_{th}$ in the rare case as depicted by Figure 4-3(b), or that $x_{i+t} > x_{th}$, which will be dealt with in Case 3. A third possibility is that $x_{i+1} < x_{th}$ and we remain in Case 1. If, however, the queue is not empty when the single-task threshold policy is in use, then a task would be allocated to the server when its state is $\leq x_{th}$, hence momentarily reducing the queue size by 1.

This case, however, is classified to be uncontrollable since in the case that the queue is empty, there exists no method of control to prevent the excessive periods of idleness. Since the server can only be allocated tasks currently in the queue and since there exists only a single arrival process of tasks into the queue, we hence have no means to allocate a task to the server even if the server state equals x_{th} if there exists no tasks in the queue.

2. $s_{stoc}(x_{th}) = \mathcal{S}(x_{th})$: This case is similar in behavior to when the server is deterministic. In this rare event that the time taken to service a task is exactly



(a) Illustration that $x_1 < x_{th}$. The grey line indicates the desired behavior given a deterministic server while the black line indicates the behavior of a stochastic server when $\overline{s_{stoc}(x_{th})} < \overline{\mathcal{S}(x_{th})}$



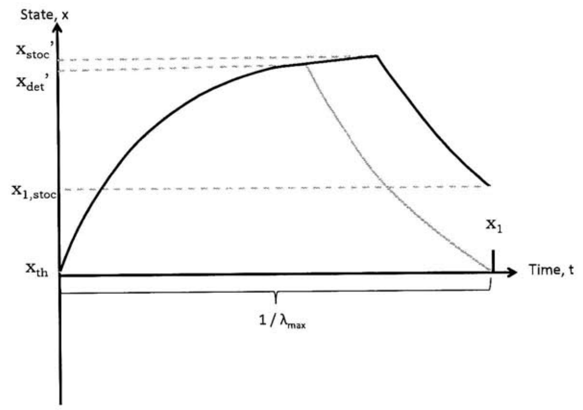
(b) Illustration of Stabilizing Property. A shorter than expected service time for a task results in an eventual increase in the mean service time of the next task to avoid excessive periods of idleness

Figure 4-3: Illustration of state behavior under single-task threshold policy and when operating in Case 1

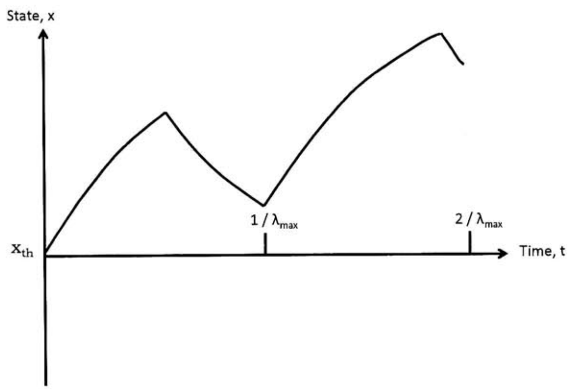
equal to what is desired, then $x_1 = x_{th}$ by definition since the total excursion time is exactly equal to $\frac{1}{\lambda_{max}}$.

3. $s_{stoc}(x_{th}) > \mathcal{S}(x_{th})$: In this third case, $x_1 > x_{th}$ if no policy is applied since the server is not given sufficient amount of idle time for its state to return to x_{th}. This is shown in Figure 4-4(a). This case is termed to be unstable if left on its own but controllable otherwise. This is because since $x_1 > x_{th}$, and given that no control is implemented, $\overline{\mathcal{S}_{stoc}(x_1)} > \overline{\mathcal{S}_{stoc}(x_{th})}$. The latter statement follows from Lemma 4.1.1 and will be elaborated on subsequently. Hence, when no control is applied, the server state at which tasks are allocated gradually increases, resulting in a corresponding increase in the mean service times of each task. This vicious cycle inadvertedly results in the server state asymptoting to 1, hence indicating the instability, as depicted by Figure 4-4(b). During this process, however, it should be noted that the queue size grows as well since the service rate is now lower than the arrival rate.

It should be apparent then that the queue size would likewise grow under the single-task threshold policy. This is because since the server state $\frac{1}{\lambda_{max}}$ time



(a) Illustration that $x_1 > x_{th}$. The grey line indicates the desired behavior given a deterministic server while the black line indicates the behavior of a stochastic server when $\overline{s_{stoc}(x_{th})} > \overline{\mathcal{S}(x_{th})}$



(b) Illustration of Instability. If $s_{stoc}(x_{th}) > \mathcal{S}(x_{th})$, the subsequent server state at which the next task is allocated will be $> x_{th}$, resulting in a corresponding increase in the expected service time for the next task

Figure 4-4: Illustration of state behavior under single-task threshold policy and when operating in Case 3

units after the allocation of a task is greater than x_{th}, and since tasks are only allocated to the server when its state is $\leq x_{th}$, then arriving tasks will continually accumulate in the queue as they are withheld from the server until the server's state decreases to x_{th}. Hence, in the event that the server continually takes a longer amount of time than desired to service the tasks allocated to him, then the queue size would resultantly grow.

However, this case is considered controllable since given that the total excursion time required would be greater than each interarrival interval, a task allocation policy like the single-task threshold policy can hence be implemented. As mentioned in Chapter 3, such policies typically are concerned with controlling the server state by scheduling the beginning of service of tasks after their arrival and are unable to interfere with the server while it is servicing a task. Hence, under the single-task threshold policy, we note that since the total excursion time required in this case would always be greater than each interarrival interval, the queue length would grow. The presence of tasks in the queue hence allow a policy to be implemented such that tasks can either be allocated or withheld from the server, unlike in Case 1, where tasks could only be withheld since the queue length is always empty.

Lemma 4.1.1. *Given that $\overline{S_{stoc}}(x)$ is convex with respect to x, and that $x_{th} = \arg\min_x\{\overline{T(x,1,\tau)}\}$, where $\overline{T(x,1,\tau)}$ is the mean excursion time for state x under a single-task threshold policy, then it can be said that $x_{th} \geq x_{min}$, where $x_{min} = \arg\min_x\{\overline{S_{stoc}(x)}\}$.*

Proof. The proof for this lemma is simple and is a result of the the convexity of $\overline{S_{stoc}}(x)$. A necessary condition for x_{min} to be the argument that minimizes $\overline{S_{stoc}(x)}$ is that $\frac{d\overline{S_{stoc}(x)}}{dx} = 0$. Given that $\overline{S_{stoc}(x)}$ is convex, there will exist only one global minimum. Also, given that the excursion time is defined to be the sum of the service time and the amount of idle time needed to return the server state back to x_{th} under a single-task threshold policy, we can then use its first derivative to determine properties of x_{th}. Note that the result found earlier Lemma 3.3.1 can be extended in this case to prove that the $T(x, k, \tau)$ is convex for all k if $\overline{S_{stoc}}(x)$ is convex, and hence will not be dealt with again.

Hence, given that both $\overline{S_{stoc}}(x)$ and $\overline{T(x,1,\tau)}$ are convex, let us examine the derivative of the latter at x_{min}, that is when $\frac{d\overline{S_{stoc}(x)}}{dx} = 0$.

$$\overline{T(x,1,\tau)} = \tau \cdot \ln(\frac{1-(1-x)e^{-\frac{\overline{S_{stoc}(x)}}{\tau}}}{x}) + \overline{S_{stoc}(x)}$$

$$\frac{d\overline{T(x,1,\tau)}}{dx} = \frac{\tau}{1-(1-x)e^{\frac{-\overline{S_{stoc}(x)}}{\tau}}} \cdot [x(1-x)\frac{\overline{S'_{stoc}(x)}}{\tau}e^{\frac{-\overline{S_{stoc}(x)}}{\tau}} + e^{\frac{-\overline{S_{stoc}(x)}}{\tau}} - 1]$$

$$\frac{d\overline{T(x_{min},1,\tau)}}{dx} = \frac{\tau}{1-(1-x_{min})e^{\frac{-\overline{S_{stoc}(x_{min})}}{\tau}}} \cdot [e^{\frac{-\overline{S_{stoc}(x_{min})}}{\tau}} - 1] < 0$$

Hence, as illustrated in Figure 4-5, since the first derivative of $T(x, 1, \tau)$ is negative when $x = x_{min}$, $x_{th} > x_{min}$. Note that $x_{th} = x_{min}$ only in the case that $x_{min} = 1^-$ due to the limits on x. This is an important property in analyzing Case 3 previously since it implies that accepting any task at a server state $x_i > x_{th}$ would imply that the subsequent mean service time of that task would be longer than desired. Note that the converse is not true and accepting any task at a server state $x_i < x_{th}$ need

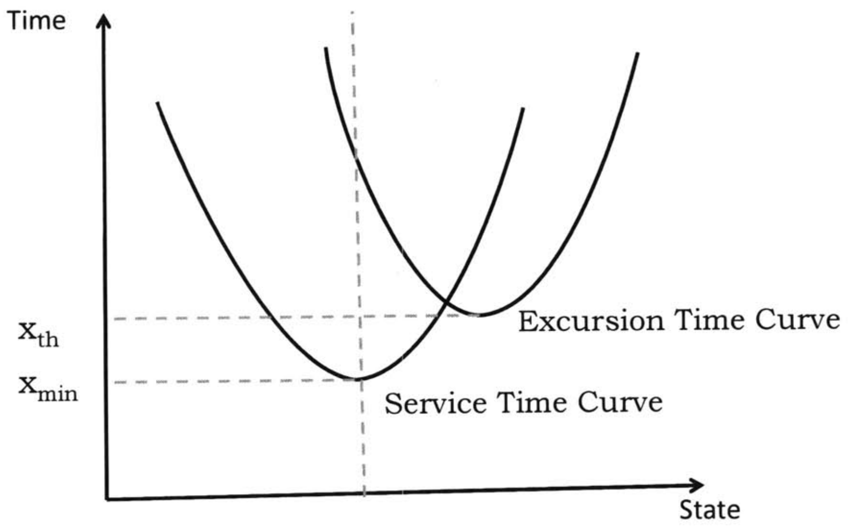


Figure 4-5: Illustration of how the service time and excursion time curves may vary with server state, x

not necessarily imply that the subsequent mean service time of that task would be longer than desired. This is hence the distinction between service times and excursion times and as has been shown before in Chapter 3 and will be shown again later, useful policies typically require the minimization of the latter and not the former. $\square$

4.2 Derivation of Bounds for the Maximally Stabilizable Arrival Rate for a Stochastic Server

In this section, we determine an upper bound for the maximally stabilizable arrival rate for a stochastic server. To do so, we first examine what we know of this arrival rate. First, similar to what was argued in Chapter 3, the lower bound of the maximally stabilizable arrival rate for a stochastic server occurs in the case when the server operates with no control and is found to be $\frac{1}{\mathcal{S}_{stoc}(1^{-})}$. This happens when the server is continually kept busy, hence resulting in the server state asymptoting to 1. Since a maximally stabilizable arrival rate refers to an arrival rate that ensures the queue is kept stable, that is that the queue size remains finite even as time tends to infinity, we can then treat this similar to a D/M/1 queue and the mean service rate must hence be at least equal to the arrival rate for the queue to remain stable.

Next, we attempt to determine an upper bound for the maximally stabilizable arrival rate for a stochastic server. As expected, such an arrival rate has to be used in conjunction with some task allocation policy. Let us first re-examine the behavior of a stochastic server under a single-task threshold policy proposed in Chapter 3 for a deterministic server with an initially empty queue. As mentioned in the earlier section, we note that the queue length decreases by 1 with a lower bound of 0 when the actual service time is less than what is desired and increases by 1 when the actual service time is greater than what is desired. The desired service time is $\overline{\mathcal{S}_{stoc}(x_{th})}$, which is the mean of the random variable $\mathcal{S}_{stoc}(x_{th})$, whose probability density function is expected to follow a lognormal distribution as shown in Figure 4-1. Hence, if we are to follow a similar argument to that of a D/M/1 queue and given that we know that a single-task threshold policy outperforms any other n-task threshold policy where $n > 1$ as shown in Chapter 3, then we note that a strategy to determine the maximally stabilizable arrival rate would then be to ensure that the mean excursion time at x_{th} is equal to the interarrival time between consecutive tasks.

First, let us say that the mean excursion times under a single-task policy, $\mathcal{T}(x, 1, \tau)$ are related to the state according to Equation 4.1.

$$\begin{aligned} \mathcal{T}(x,1,\tau) &= \int_0^\infty [\tau \ln(\frac{1-(1-x)e^{-t/\tau}}{x}) + t] \cdot f(x,t)dt \\ f(x,t) &= \frac{1}{t\sqrt{2\pi\sigma^2(x)}} e^{\frac{-(\ln t - \mu(x))^2}{2\sigma^2(x)}} \end{aligned} \tag{4.1}$$

where $f(x,t)$ is the probability density function of t at state x. Hence, if we define $x_{th,stoc}$ to be the argument that minimizes the mean excursion time under a single-task policy as shown in Equation 4.2, then the upper bound on the maximally stabilizable arrival rate is hence $\frac{1}{\mathcal{T}(x_{th},1,\tau)}$.

$$x_{th,stoc} := \arg\min_x \{\mathcal{T}(x,1,\tau)\} \tag{4.2}$$

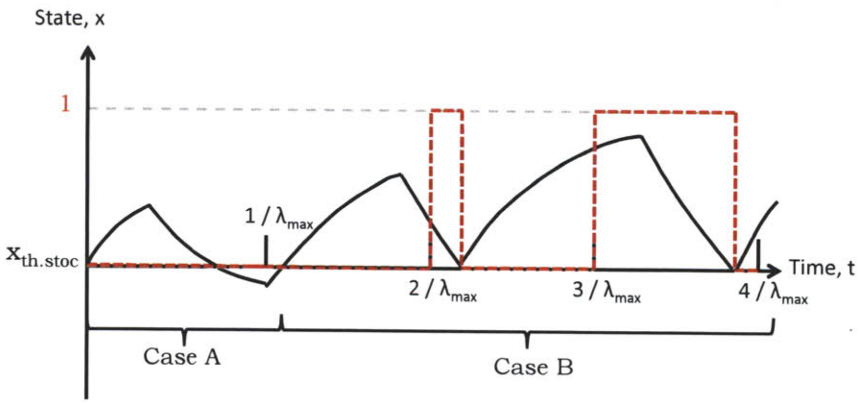


Figure 4-6: The black line in this figure illustrates how the server state could vary under a single-task threshold policy. Two possible cases are expected under this policy, the first of which is when a task is assigned at a state below $x_{th,stoc}$ (Case A) and the second is when a task is assigned at precisely $x_{th,stoc}$ (Case B). The red line in the figure indicates the size of the queue as time progresses.

4.2.1 Upper Bound Analysis

Though we find the upper bound to be $\frac{1}{\mathcal{T}(x_{th,stoc},1,\tau)}$, from our analysis, we find that there exist certain conditions in order for this to hold true. First, it is important for readers to note that the upper bound was determined by requiring the interarrival times of new tasks to the system to be equal to the mean of the excursion time for a single-task threshold policy. Note that the single-task threshold policy is once again treated as the optimal policy among the class of all stationary policies since Lemmas 3.3.2 and 3.3.4 still hold true when considering the means of the excursion times.

Moreover, a D/M/1 queue is stable if the arrival rate is equal to the mean service rate since by the strong law of large numbers, the sample average of the service times converge almost surely to the mean service time, implying that tasks are serviced as fast as they enter the system. Hence, though the queue size need not necessarily remain at 0 throughput, the queue is stable in the long-run and the queue length will never diverge to ∞. However, in this case, though we attempt to apply the same argument to the excursion times, it should be apparent that this does not hold true, as can be seen from Figure 4-6.

An excursion from state x is defined to be the event that starts with the acceptance

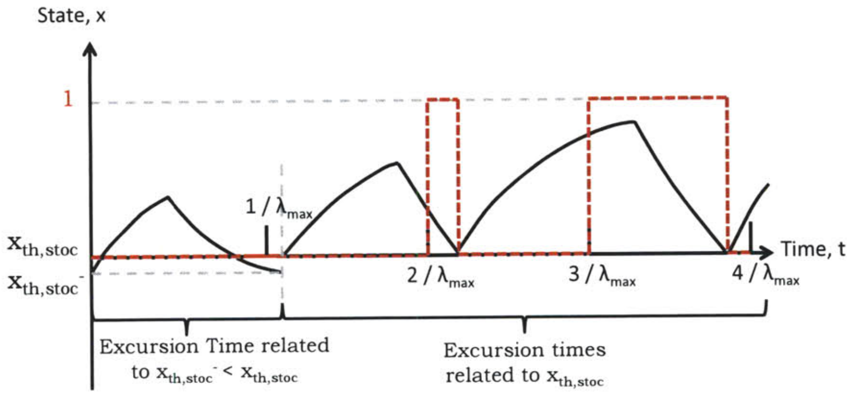


Figure 4-7: Illustration of how assigning a task at $x_{th,stoc}^- < x_{th,stoc}$ results in an excursion time associated with a higher mean excursion value, $\mathcal{T}(x_{th,stoc}^-, 1, \tau)$. This follows from Lemma 3.3.3.

of a task by the server at state x and ends with the first subsequent return to state x. From Figure 4-6, we note that in Case B shown, when a task is assigned at precisely state $x_{th,stoc}$, then the mean of the associated excursion times is expected to converge to the mean excursion time, $\mathcal{T}(x_{th,stoc}, 1, \tau)$. However, it is also apparent that if a task is ever assigned at a state less than $x_{th,stoc}$, then the excursion time associated with the previous task can be said to be a sample from the random variable $\mathcal{T}(x_{th,stoc}^-, 1, \tau)$, where $x_{th,stoc}^-$ is the state that the task was assigned at and is $< x_{th,stoc}$, which follows from Lemma 3.3.3. This is illustrated in Figure 4-7. Such a scenario will occur almost surely if the queue were empty to begin with, hence implying that the true maximally stabilizable arrival rate should be even less than expected, since $\mathcal{T}(x_{th,stoc}^-, 1, \tau) > \mathcal{T}(x_{th,stoc}, 1, \tau)$ by definition. Hence, the single-task threshold policy with an arrival rate of $\frac{1}{\mathcal{T}(x_{th,stoc}, 1, \tau)}$ should be applied to a non-empty queue, for example by allowing a buildup of a certain number of tasks in the system before the first task is assigned. A slightly lower arrival rate, however, will guarantee stability for a stochastic server even if the queue is empty to begin with.

4.3 Simulation and Conclusion

In this section, we present some simulation results for a server with stochastic service rates, where the mean service times vary with state according to the Yerkes-Dodson law, while the distribution of the service times for a given state follow a lognormal distribution. An illustration is shown in Figure 4-2. The parameters used in the simulation are summarized below.

$$\text{Mean Service Time,} \quad \overline{\mathcal{S}_{stoc}(x)} = 229x^2 - 267x + 99$$

$$\text{PDF of Service Time at state } x, \quad f(x,t) = \frac{1}{t\sqrt{2\pi\sigma^2(x)}} e^{\frac{-(\ln t - \mu(x))^2}{2\sigma^2(x)}}$$

$$\text{Mean Excursion Time,} \quad \mathcal{T}(x,1,\tau) = \int_0^\infty [\tau \ln(\frac{1-(1-x)e^{-t/\tau}}{x}) + t] \cdot f(x,t)dt$$

$$\text{Stand. Dev. of Excursion Time,} \quad \sigma = 10$$

$$\text{Sensitivity Parameter,} \quad \tau = 300$$

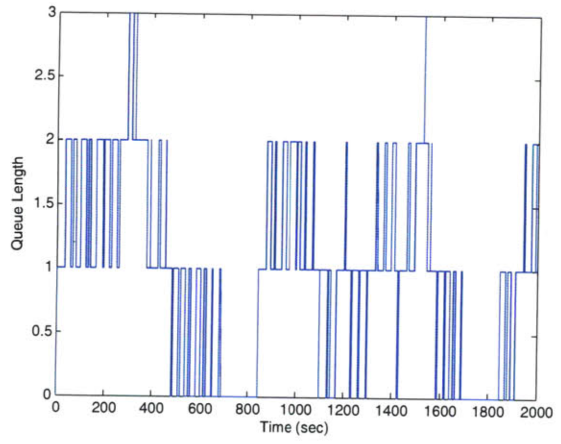
$$\text{Calculated Threshold Value,} \quad x_{th,stoc} = 0.654$$

$$\text{Upper Bound on } \lambda, \quad \lambda_{max} = 0.031$$

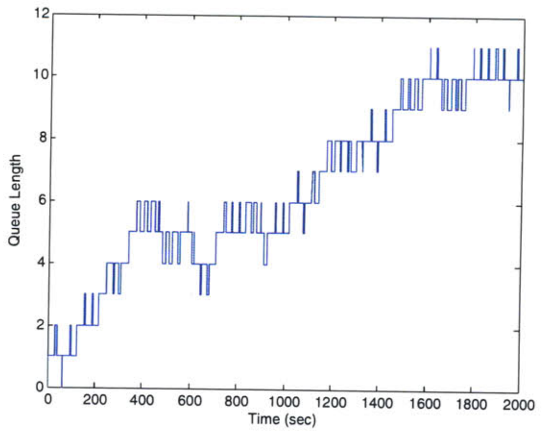
$$\text{Multiplication Factor,} \quad \epsilon = 0.05$$

Figure 4.3 compares the queue length when $\lambda = \lambda_{max}$ and when $\lambda = (1+\epsilon)\lambda_{max}$. As can be seen, the queue length remains stable for the former case but grows for the latter case. This proves the validity of the upper bound.

In conclusion, Chapters 3 and 4 provide detailed analysis on optimal policies and upper bounds on the maximally stabilizable arrival rates for deterministic and stochastic servers respectively. In the former case, the optimal policies were found for objective functions that seek to maximize the total useful throughput and we found these policies to be extendible to all cases, regardless of whether the objective functions were convex with respect to x. In the latter case, however, we were only able to formulate bounds on the maximally stabilizable arrival rate for the purpose of maximizing the total throughput (that is, error rates are not considered) and these bounds are only valid for the case that the objective function is convex. Chapter 4



(a) Queue size when $\lambda = \lambda_{max}$



(b) Queue size when $\lambda = (1+\epsilon)\lambda_{max}$

Figure 4-8: Comparison of Queue Lengths for Different Deterministic Arrival Rates

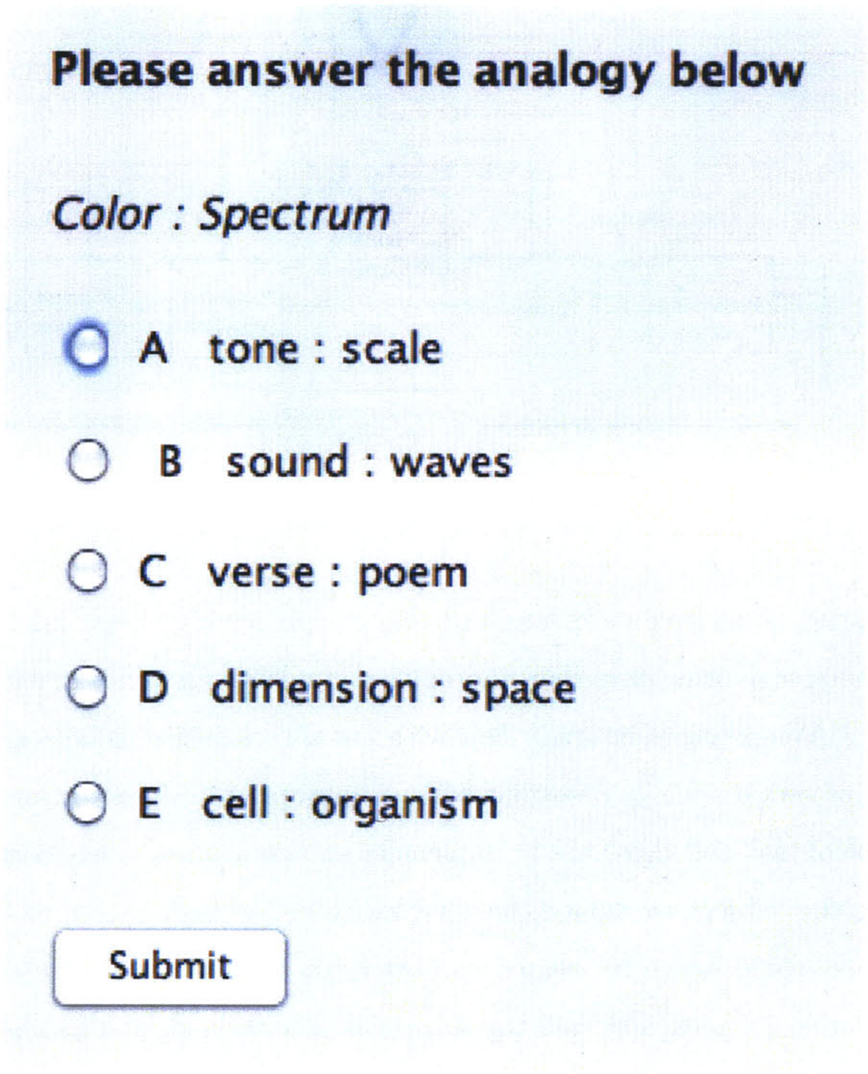
is hence most definitely not an exhaustive analysis of the behavior of a stochastic server. In the subsequent chapters, however, we will attempt to implement the policies introduced in Chapters 3 and 4 in a real-life setting and examine if such policies are in fact feasible in reality. Difficulties in estimation of the sensitivity parameters and/or the server state as well as lessons learnt will be discussed in the following 2 chapters.

Chapter 5

Experimental Design

This chapter discusses the experimental design used for the implementation and assessment of the single-task threshold policy introduced in Chapter 3 in order to maximize the total useful throughput of a human. As mentioned in Chapter 2, we envision such policies to eventually be used in the realm of Human Supervisory Control, for example by UAV operators involved in reconnaissance missions. Tasks such as gathering of on-site information using a UAV or target classification from the images streamed from a UAV's camera out in the field would then make up the types of tasks that are continually sent to the UAV operator and the goal of the task scheduler would then be to allocate such tasks so that the number of tasks done correctly by the operator is maximum.

One should take note, however, that it is not claimed that these experiments provide conclusive information of the single-task threshold policy. Instead, these should be viewed as pilot studies to investigate the feasibility of such policies and to examine difficulties in implementation. Moreover, theoretical policies may work on paper but actual implementation may be difficult and the results may not be according to what was predicted. This chapter hence summarizes the pilot study we undertook to test the feasibility of the policy developed in Chapter 3 and the subsequent chapter will discuss more on the results.


Please answer the analogy below

Color : Spectrum

- A tone : scale
- B sound : waves
- C verse : poem
- D dimension : space
- E cell : organism

Figure 5-1: Sample Graphical User Interface used for Experiments

5.1 Design of Experiment

Figure 5-1 shows an example of the experimental interface, with the question shown and the 5 possible answers to the question listed in bullet-form below. The type of task we chose for the pilot experiments was verbal analogy, similar to those often posed to examinees on the Scholaristic Assessment Test (SAT). This task was chosen as it is classified as a cognitive task that requires subjects to both decipher the meaning of words, form relations between the words and to make decisions on which option is the correct answer, where the correct answer is the one that has the closest relation to the

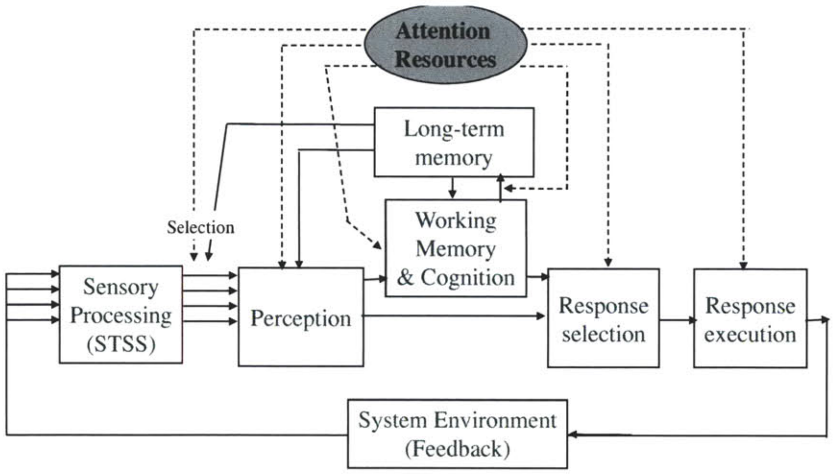


Figure 5-2: Human Information Processing Model proposed by Wickens [6]

posed question. Cognitive tasks hence involve the processing of new information and the ability to recall or retrieve that information at a later time (from memory). Since tasks that UAV operators typically deal with are also classified to be cognitive tasks, the choice of verbal analogy questions seemed appropriate. Moreover, we opted for a simple type of task and chose not to implement an actual supervisory control mission setting for the subject to reduce the complexity of the task shown to the subject. Performance would hence be related to how much time the subject spends solving the task, rather than on how well the subject is able to learn and handle a complex mission during the experiment. A model of the human information processing model is shown in Figure 5-2.

A minimalistic style was also chosen to present the task to the subject. In fact, all that was shown to the subject was the analogy question and 5 possible answers to the question, with no feedback of any kind on their performance. We opted not to show any indication of the amount of time spent or time remaining in the experiment to avoid having any factor (apart from workload) pressurizing the subject. The number of tasks in the queue and feedback on the subject's performance were hence not shown as well. Questions were chosen and distributed in a manner such that there was no bias in the difficulty of the questions at any point in the experiment. Additionally, the questions were always posed to the subjects in the same order to avoid inconsistency. Subjects were also not given the option to go back and change their answers upon submission and were also always required to provide an answer with no option to skip questions.

Before the experiment began, subjects were also briefed on their objectives. Subjects were all told to maximize the number of correct answers while answering as many questions as quickly as possible. Subjects were told that they would be awarded a point for every correct answer, but also penalized with a subtraction of a point for every incorrect answer. An assumption made during these experiments was then that subjects understood and followed these objectives thoroughly and that they were always trying to maximize their score. The associated predicted behavior of such a subject would hence be one where he answers questions that are obvious quickly, while deliberating for a longer time on questions that are not as obvious. During this period, the subject is then making a tradeoff between answering the question correctly or answering more questions within the allotted timeframe.

5.2 Implementation of the Task Allocation Policy

The task allocation policy implemented in the experiment is the single-task threshold policy proposed in Chapter 3 and it is restated here for completeness.

$$u_{\text{TP}}(t) = \begin{cases} \text{ON} & \text{if } x(t) \leq x_{\text{th}}(\tau), \\ \text{OFF} & \text{otherwise,} \end{cases}$$

However, though this policy appears simple, there exist many system parameters such as the initial server state $x_0(0)$ and τ that are unknown to the task controller at the start of the experiment. Furthermore, parameters such as τ would probably differ from person to person since it is a measure of how sensitive the subject is to sudden changes in workload. Moreover, the service time and error rate functions also need to be derived during the experiment since they differ from person to person. The remainder of this section hence summarizes the methods we used to obtain these parameters, as well as some suggestions on how these methods could be improved.

5.2.1 Estimation of Initial State and Sensitivity Parameter

The first step to be taken by the controller is to estimate the initial state and sensitivity parameter of the subject. The variation of the server state with time for the same busyness profile under different initial states and sensitivity parameters are shown in Figures 2-2 and 2-3 respectively. The choice of x_0 was eventually decided to be arbitrarily set to 0.5 since it is apparent from Figure 2-2 that given the same busyness profile, the server state will eventually converge regardless of the initial state. Since $x \in (0, 1)$, choosing the midpoint seemed appropriate. Alternative methods of deriving this initial state could include more subjective measures such as asking the subject to input how busy he feels at the start. This, however, does have drawbacks since it is a subjective measure and it might be hard to ask a subject to evaluate his busyness when he has yet to start working on a task.

Next, the sensitivity parameter was chosen to be $\tau = 150$. The resultant choice of this value was based on a tradeoff between the amount of time it takes for the eventual convergence of the state and the realism of the model. This is because small values of τ were found to result in faster convergence rates of the state for varying initial values. Small values of τ also allowed the state to change more drastically, which is important in our modelling given that we do not have the luxury of running experiments that are hours-long. However, excessively small τ values would also imply that humans are only sensitive to how busy they have been in the very recent past, hence resulting in huge fluctuations in the server state, which do not appear realistic. The value of 150 was hence chosen since simulations based on pilot studies showed it to be a suitable choice to model the server state, particularly since previous work in [3] had used the same value in the calculation of the server workload using a moving average formula.

5.2.2 Determination of $\mathcal{S}(x)$ and $\mathcal{E}(x)$ Functions

After deciding on x_0 and τ, we can then determine the $\mathcal{S}(x)$ and $\mathcal{E}(x)$ functions for each subject. This was done by implementing a threshold policy with three different threshold values over 30 minutes. The threshold values chosen were $x_1 = 0.6$, $x_2 = 0.7$, and $x_3 = 0.8$ and tasks were assumed to be always available in the queue during this period. One can then take an average of the service times and error rates associated with each threshold value, with some tolerance allowed to ensure a sufficiently large dataset. Using these mean values and their associated x_i values, one can then do a least-squares fit to obtain an approximation of $\mathcal{S}(x)$ and $\mathcal{E}(x)$. The assumption made here is hence that both $\mathcal{S}(x)$ and $\mathcal{E}(x)$ are convex with respect to x and that both follow a second-order polynomial. The curve-fitting was done using QR decomposition that is part of the JAMA package in JAVA. Finally, it should be noted that the 30 minute period was not divided equally between each threshold value. This is because as is characteristic of all exponential variables, the server state increases at a decreasing rate as the state increases from 0 to 1, since the state asymptotes to 1. Hence, the amount of time, t_{01}, needed for the state to increase from $x_0 = 0.5$ to $x_1 = 0.6$ will always be less than t_{12} and t_{23}. Since $t_{01} \leq t_{12} \leq t_{23}$, the amount of time allocated to the first threshold was 300 seconds. The time allocated to collect readings for the second threshold was 700 seconds while the time allocated for the third threshold was 800 seconds.

Alternative methods to derive these functions include binning the service times and error rates into arbitrarily sized Δx-intervals and finding the means related to each bin. A curve-fit can then be done to determine the curves. This is similar to what was done in [19]. Problems faced when using this method include difficulties in separating transients from useful experimental data and the high sensitivity of the data to bin-size. The first problem arises due to the need to determine how long a transient to neglect before the data is collected. This problem is avoided by the former method since data is only collected when the server state equals to x_i and hence the transient period is ignored. The latter problem, on the other hand, arises due to the lack of sufficiently large datasets, especially for lower values of x, resulting in inaccurate mean values for the left-hand side of the curve. This has been a long-standing problem of characterizing the Yerkes-Dodson law and hence we mitigate this by purely finding the mean values at 3 threshold values, since 3 is the minimum number of datapoints needed to fit a second-order polynomial. While

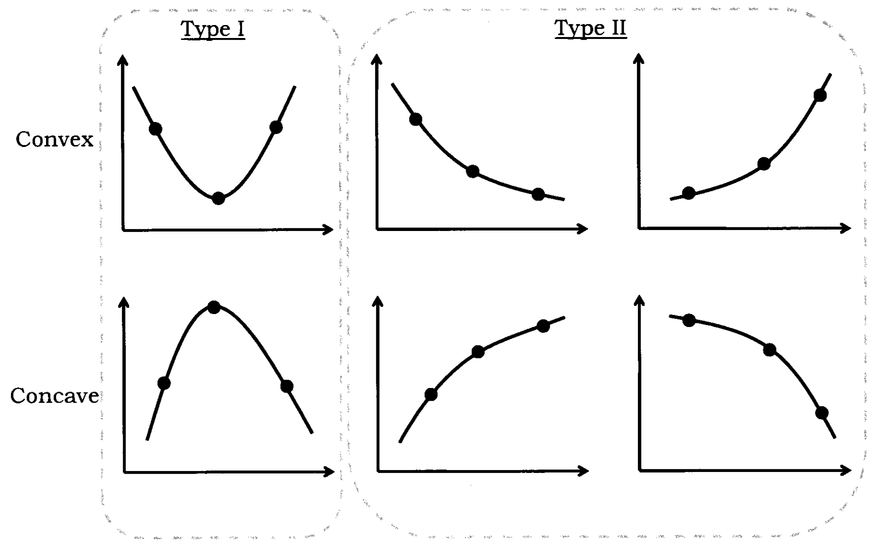


Figure 5-3: Classification of Possible Types of Curves Obtained for $\mathcal{S}(x)$ and $\mathcal{E}(x)$

it is tempting to increase the number of threshold values, one must note that each subsequent increment requires an exponential increase in the amount of time needed to collect readings. An overly-long experiment would most probably not be welcomed by subjects and factors such as boredom and fatigue could set in.

5.2.3 Classification

After the first 30 minutes of the experiment and with the $\mathcal{S}(x)$ and $\mathcal{E}(x)$ functions determined, we then need to determine the x_{th} value. While it was proposed in Chapter 3 that each subject's unique objective function would be a combination of both minimizing his service times and error rates, determining the weightage that each subject places on each factor is difficult. Hence, for our pilot study, we avoided this problem by simply deciding to focus on either the service times or the error rates for each subject. This was done by classifying each subject to either be service-time-sensitive (Case 1), error-rate-sensitive (Case 2), or neither (Case 3). Future work can then focus on determining methods to decipher each individual's preferences.

Subjects are classified using the following method. First, as shown in Figure 5-3, each function is classified to be either convex or concave. In the event that $\mathcal{S}(x)$ is convex and $\mathcal{E}(x)$ is concave, then the subject is determined to be service-time-sensitive and vice versa. However, in the event that both $\mathcal{S}(x)$ and $\mathcal{E}(x)$ are convex, then the subject is classified according to the function with the steeper average 'gradient'. A subject is termed to be 'uncontrollable' in the event that both $\mathcal{S}(x)$ and $\mathcal{E}(x)$ are concave. Implementing a threshold controller would not be useful to the subject in this case and the subject is better left to his own devices.

The average 'gradient' for the service time function and the error rate functions are defined to be $\overline{\triangle\mathcal{S}}$ and $\overline{\triangle\mathcal{E}}$ respectively and are calculated as shown in Equation 5.1. Figure 5-4 illustrates how $\triangle\mathcal{S}_1$ and $\triangle\mathcal{S}_2$ can be calculated if the penalty measure were the service time.

$$\begin{aligned}
\triangle\mathcal{S}_1 &= \mathcal{S}(0.6) - \mathcal{S}(0.7) \\
\triangle\mathcal{S}_2 &= \mathcal{S}(0.8) - \mathcal{S}(0.7) \\
\overline{\triangle\mathcal{S}} &= (|\triangle\mathcal{S}_1|/\mathcal{S}(0.7) + |\triangle\mathcal{S}_2|/\mathcal{S}(0.7))/2 \\
\\
\triangle\mathcal{E}_1 &= \mathcal{E}(0.6) - \mathcal{E}(0.7) \\
\triangle\mathcal{E}_2 &= \mathcal{E}(0.8) - \mathcal{E}(0.7) \\
\overline{\triangle\mathcal{E}} &= (|\triangle\mathcal{E}_1|/\mathcal{E}(0.7) + |\triangle\mathcal{E}_2|/\mathcal{E}(0.7))/2
\end{aligned} \tag{5.1}$$

Finally, upon determining if the subject is service-time-sensitive or error-rate-sensitive, the final step would be to determine x_{th}. Determining this value, however, requires one final classification of the type of convex graph. We propose there to exist 2 types of convex graphs as shown in Figure 5-3: Type 1 is when the minimum point of the graph lies within the range (0,1), $x_{min} \in (0,1)$ while Type 2 is when $x_{min} \notin (0,1)$. Determining x_{th} for Type 2 convex graphs is easy and $x_{th} = \arg\min_x \mathcal{S}(x)$ in the event that the subject is service-time-sensitive. In our case, however, we opt to restrict $x_{th} \in \{0.6, 0.7, 0.8\}$.

Determining x_{th} for a Type 1 convex graph is, however, not as straightforward. Assuming that a subject is found to be service-time-sensitive, and that his service time function can be classified to be a convex Type 1 convex graph such that $\mathcal{S}(x) = a_s x^2 + b_s x + c_s$, then we find that $x_{th} \approx \sqrt{\frac{c_s}{a_s}}$. On the other hand, if the subject is

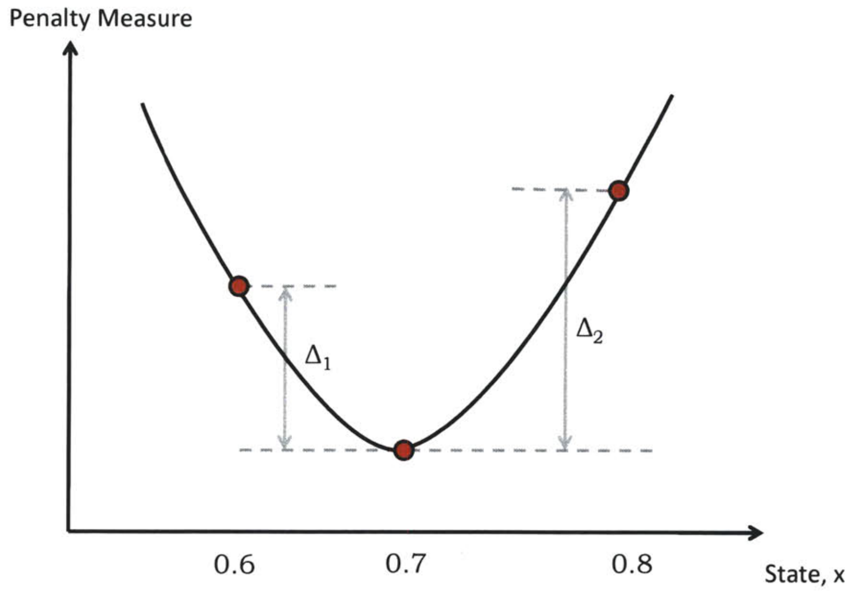


Figure 5-4: Illustration of how the $\triangle_1$ and $\triangle_2$ values are calculated

error-rate-sensitive such that $\mathcal{E}(x) = a_e x^2 + b_e x + c_e$, then $x_{th} = \frac{-b_e}{2a_s}$.

5.3 The Experiment

The experiment was designed to last at most 45 minutes. Participants signed an informed consent form, and filled out a demographic survey prior to the start of the experiment. This survey requested for information such as age, a self-evaluated level of command of the English language, and the number of years the participant has been communicating in English. Next, a simple training session was done, which lasts about 5 minutes. This consisted of subjects reviewing training slides, which included a brief introduction to the experiment and the type of questions that they would be posed since it is imperative that subjects understand how to solve analogy questions before they are even allowed to begin the experiment. Finally, the slides also included the objective that the subjects were to maximize and presented a scoring system of how their performance would be evaluted. The scoring system was the same for all participants and they were told that they would be given 1 point per correct answer, though 1 point would be subtracted from their score for an incorrect answer. Participants, however, were not informed that there existed a control policy working 'behing-the-scenes' that was allocating when they would receive a task so as to avoid biasing their responses. Participants also were also not informed about how the experiment would be carried about and instead were simply instructed to maximize their objective and answer the questions as they were given to them. The use of control hence represents the independent variable of the experiment.

As mentioned earlier, the experiment began with a single-task threshold policy implemented at $x_1 = 0.6$ for 300 seconds. This was immediately followed by a threshold policy implemented at $x_2 = 0.7$ and $x_3 = 0.8$ for 700 and 800 seconds respectively. This was done to collect data associated with each threshold value and these data hence aided in determining the $\mathcal{S}(x)$ and $\mathcal{E}(x)$ functions. Each subject was then judged to be service-time-sensitive or error-rate-sensitive and the threshold value, x_{th} was then calculated based on the associated curve. All this happened behind the scenes and was unknown to the subject.

Finally, the remaining 15 minutes of the experiment was concerned with the subject's overall performance given an arbitrarily set arrival rate of $\lambda = 0.1$ and with a single-task threshold policy with threshold value x_{th} implemented. After the experiment, the participants were then given a final survey in which he was asked several subjective questions including rating the intensity of the experiment and his thoughts on how he could have performed better in the experiment. Final interviews were then done to debrief the subject.

5.4 Variables and Measurements

The only independent variable in this experiment is the use of a control policy. However, one should notice immediately that our experiment did not include a period of testing with no control. The decision to not include such a testing period was due to time limitations during the actual experiment, since adding in such a period would prolong an already lengthy 45-minute experiment. Nevertheless, there is still basis for comparison of the feasibility of the single-task threshold policy by comparison with the performance at the 3 threshold values of 0.6, 0.7 and 0.8.

Finally, the dependent variables measured in this case were the service times and error rates and were observed by recording data during the experiment.

Chapter 6

Experimental Results

This chapter presents the analysis of the data collected from the experiment. A summary of the experimental proceedings is first provided, followed by the experimental results and our analysis. The last section then revisits the data and provides discusses possible reasons for the results obtained as well as common behaviors observed during the experiment.

6.1 Experimental Set-up

There were 26 participants, 10 male and 16 female, in the experiment for this thesis. Subjects were recruited through word-of-mouth and mailing lists and experiments were conducted both on-campus and online. 6 experiments were conducted at the Behavioral Research Laboratory on the MIT campus while the remaining 20 experiments were conducted using online methods. In the latter case, the experiment software along with the instructional slides and surveys were disseminated through email contact with each participant and effort was made to ensure that subjects in both cases had the same level of instruction on what was required of them, particularly with regards to the objective function they were to maximize.

Recruitment of participants was biased towards English speakers since a good grasp of English is required for the test. However, a good command of vocabulary was not a requirement since we did not expect participants to recognize all the words in the test, but instead simply expect them to spend time thinking. Moreover, results from the demographic survey showed that while English was the primary spoken language among the participants, most of them rated their command of English to only be a 3 out of a scale of 5. Most participants also had studied for or taken the Verbal Scholaristic Assessment Test previously, though none admitted to having a good grasp of answering analogy questions.

6.2 Experiment Result Tabulation and Analysis

Of the 26 participants tested, it was found that only 16 participants had the single-task threshold policy implemented. This implies that the remaining 10 participants either had concave curves for both their service time and error rate functions or that the participants took extremely long (up to hundreds of seconds per question) for each question, resulting in there being insufficient data to determine their service time and error rate functions. The remainder of this analysis hence concentrates on the data from the 16 participants.

Results from the 16 participants who could be classified indicated that only 3 of the 16 participants were found to be error-rate-sensitive. The remaining 13 participants were found to be service-time-sensitive. This result is similar to the results obtained from our preliminary testing and previous tests and possible reasons for this will be given in the subsequent section. In addition, 14 of the 16 participants were also found to exhibit Type I behavior, meaning that the minimum point of their associated convex graph lies within the range $x = (0, 1)$. This once again agrees with what was found in our preliminary experiments. Finally, the remaining 2 participants in the latter case had functions that were either strictly increasing or decreasing within the range $x = (0, 1)$.

Table 6.1 summarizes the service time behavior of the 13 participants who were found to be service-time-sensitive. It is apparent that the implementation of the threshold policy does result in faster service times in general. As mentioned in Chapter 3, since the performance of a server without any control can be expected to be

Subject No.	x_1 **= 0.6**	x_2 **= 0.7**	x_3 **= 0.8**	x_{thres}	**Percent Change in Service Time (%)**
S1	18.8	11.8	16.7	17.0	+2.1
S2	27.9	14.2	26.4	16.2	-38.8
S3	16.3	13.0	18.5	16.1	-13.1
S4	15.2	13.7	16.2	14.3	-11.8
S5	18.5	16.1	13.0	11.5	-11.9
S6	36.1	12.1	15.7	8.4	-46.3
S7	17.5	11.7	12.6	13.1	+3.7
S8	12.4	9.7	13.1	11.7	-10.5
S9	9.6	8.3	13.6	12.4	-9.2
S10	12.4	11.4	20.5	9.7	-52.6
S11	13.3	11.0	22.2	9.3	-58.1
S12	10.7	7.9	12.4	12.0	-3.2
S13	10.8	7.5	12.5	11.0	-11.7

Table 6.1: Summary of Results for Service-time-sensitive Participants

its performance at $x = 1^-$, we hence compare the service times using the single-task threshold policy and with $x = x_{thres}$ to the service times found when $x = 0.8$, which is the closest sample we have to $x = 1^-$. We notice an overall decrease in service time under the optimal policy with an average decrease of 20%, with a maximum decrease of 58% and a maximum increase of 4%. A paired t-test was performed to determine if the effect of the optimal policy was effective. This statistical analysis test was chosen since we are comparing two paired groups, that is using measurements from the same subject under 2 different conditions. Moreover, though the paired-t test is only valid for comparison of measurements drawn from Gaussian distributions, we can however use it since our samples are assumed to be drawn from a lognormal distribution. By taking the natural logarithm of each of these samples, the result is a value that follows the Gaussian distribution, which we can then compare using the paired t-test. From our analysis, the mean decrease in service time (M=3.90, SD=1.29, N=13) was significantly greater than zero, $t(12) = 3.10$, two-tail $p = 0.01$, providing evidence that the control is indeed effective in reducing the service times for service-time-sensitive subjects.

Table 6.2 on the other hand summarizes the results for the error-rate-sensitive participants. We notice that in this case, implementation of the optimal control

Subject No.	$x_1 = \mathbf{0.6}$	$x_2 = \mathbf{0.7}$	$x_3 = \mathbf{0.8}$	x_{thres}	**Percent Change in Error Rate (%)**
E1	0.2	0	0.35	0.58	+65.4
E2	0.14	0.27	0.5	0.33	-33.3
E3	0.36	0.25	0.36	0.59	+64.6

Table 6.2: Summary of Results for Error-rate-sensitive Participants

policy appears to worsen the server's behavior.

6.3 Discussion and Common Behaviors

In this section, we provide some insight on plausible reasons for some of the results and behaviors observed during the experiment.

First, from a review of the demographic survey associated with each participant, we notice that all 10 of the 26 participants who were disregarded from the data analysis due to reasons such as the lack of implementation of the control policy were subjects who participated in the experiment online. Some feedback received from these participants indicate that they were not able to focus fully during the experiment, particularly since they were often told to wait. During such periods, it is then not far-fetched to imagine participants using the time to surf the internet though they were specifically told not to. Moreover, many such subjects also felt alarmed with the amount of time they had to spend waiting, attributing it to lag or slow computer processors. This evitably affected their performances and could have caused unnecessary stress. To mitigate this, we propose that future experiments should always be conducted in a controlled setting where subjects feel pressurized to perform and under the watchful eye of an investigator, the presence of whom should discourage subjects from getting distracted. Also, this exemplifies the difficulties in implementation of such a policy since the optimal policy strictly relies on the subject being focussed on the task and always seeking to maximize his score and deviations can severely impact the validity of the policy.

Secondly, from the data gathered and the method of classification as presented in Chapter 5, we notice that a majority of participants were classified to be service-

Subject No.	**Percent Change in Service Time (%) Under x_{thres} Compared to $x_3 = 0.8$ (%)**	**Change in Error Rate (%) Under x_{thres} Compared to $x_3 = 0.8$ (%)**
S1	2.1	+0.32
S2	-38.8	-0.05
S3	-13.1	+0.35
S4	-11.8	+0.14
S5	-11.9	+0.11
S6	-46.3	+0.17
S7	3.7	+0.09
S8	-10.5	+0.02
S9	-9.2	-0.15
S10	-52.6	+0.38
S11	-58.0	+0.18
S12	-3.2	-0.14
S13	-11.7	+0.02

Table 6.3: Comparison of Percent Change in Service Time with the Change in Error Rates for Service-Time-Sensitive Subjects

time-sensitive instead of error-rate-sensitive. This indicates that a change in state will result in a greater change in service time for most participants, compared to the error rate. A possible reason for this may be that people simply take longer to process the same information when under high or low workload conditions due to stress or lack of concentration, though the amount of time spent on the task does not affect how well the person performs in the task. Instead, factors such as the ability to understand the question and answers given and to be able to find a relation between the words may affect the error rates to a greater degree for such subjects. On the other hand, subjects that are error-rate sensitive may be subjects who have a wide vocabulary and are hence able to figure out the correct answer to the analogy. Hence, by controlling the workload given to the subject, the subject is able to perform optimally, which results in an associated decrease in error rate.

Next, we notice that though the implementation of the optimal policy did result in a decrease in service times in general for service-time-sensitive subjects, this resulted in a subsequent increase in error ratesas well, as shown in Table 6.3. This is as expected as we mentioned in Chapter 5 that for the purpose of these pilot studies, we were only seeking to maximize either the subject's service time or error rates, and hence a subsequent increase in error rates due to the decrease in service time is indeed expected from the speed-accuracy tradeoff.

Finally, we conclude with a summary of common behaviors noticed in subjects and as gathered from the surveys handed out at the end of the experiment. In Chapter 5, we presented the rationale for choosing to use verbal analogy questions in our experiment. Though this task has proven to be simple to implement, the use of difficult vocabulary also implies that subjects who do not recognize the words will be clueless on how to answer the question. While there exist many methods to decipher words, for example through the use of word-roots, or by selecting an answer through the process of elimination, the type of task is indeed a hindrance to subjects with limited vocabulary. This results in subjects guessing when they do not know how to answer a question, many of whom would probably not even attempt to spend any time figuring out the answer should they not even understand the question. Locating an answer that seems to be the correct one at the top of the list of possible answers has also been a problem. This is because some subjects have been noticed to make their choice very quickly in such cases, without paying much attention to the other options

Boredom and tiredness were the main complaints during the experiment, with many subjects finding the time they spent waiting between tasks irksome. Subjects also indicated that the time spent waiting between tasks was too long in most cases. We foresee that this would indeed be an issue if such a policy is implemented in real-life since the frequent start-stop nature of the policy can be disruptive to a person's chain of thought. This, however, has been addressed in Chapter 3 by modifying the optimal policy to a dual switching threshold policy when switching and holding costs are considered. In addition, many subjects also proposed that the time to be spent waiting be shown to them so that they would be able to anticipate the next task. Other forms of information that subjects indicated they would like displayed include a timer and an indication of the number of tasks remaining. Many subjects also found the 45 minute experiment tiring and long, and suggested that their performance definitely dipped at the end of the experiment due to the length of time that they had to stay focussed.

Finally, we also noticed that many subjects tried to pace themselves such that they would spend a longer amount of time on a task if the interarrival time between that task and the previous task was long (that is if the subject was made to wait for a long time in between the two tasks). This is probably based on their assumption that the wait time was due to a long loading time and hence they tried to adjust for this themselves by spending a longer amount of time on the previous task, in the hope that they would not need to wait as much for the next task. This, however, only served to worsen their performance and increase the waiting time further, and hence proved to be a constant source of irritation for them.

In conclusion, the data gathered revealed several interesting results, and it is our hope that when combined with the lessons learnt and the discussion of common behaviors observed during the experiment, that the work in this thesis can indeed be used to improve task allocation policies in a wide variety of fields. Nevertheless, it is also important to reiterate that this experiment is simply a pilot study and further studies would probably be required to test the performance of the optimal policy in several different domain settings.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis presented task allocation policies for a single server state-dependent queue with deterministic arrival rates, where the server behavior follows certain conceptual laws. Optimal policies were derived for a server with deterministic service rates, and when the server was attempting to maximize his useful throughput. Such policies were found to be valid for all classes of objective functions, regardless if they were convex or general-shaped. On the other hand, servers with stochastic service rates were also considered, though no guarantees were given on what would be the optimal policy in such a case. Instead, an upper bound on the maximally stabilizable arrival rate for a server who is trying to maximize his total throughput was found, though this bound is only valid in certain cases. These policies and their associated assumptions hence form the key theoretical discoveries of this thesis.

In addition, a simple proof-of-concept experiment was carried out to investigate the feasibility of implementing such a policy in a real-world setting and results from the study have turned out as expected. A discussion of the methods that worked in implementing such a policy in the real-world and a summary of the lessons learnt from the experiments are hence also a key contribution of this thesis. However, the proof-of-concept experiment only served to show that the service times would decrease for a service-time-sensitive server when the optimal policy is used and this conclusion did not hold for the case that the server is error-rate-sensitive. Moreover, the decrease in service time also resulted in a corresponding increase in error rate and hence, it can be said that there still remains much to be said about this policy.

The main benefit of the work done for this thesis is hence the direction the results give for potential future work in the development of more complex task allocation policies under real-world scenarios. Lessons learnt from the implementation of the single-task threshold policy in a real world setting and the feedback obtained will also help guide the research to be both theoretical and realistic.

7.2 Future Work

For the future, we can foresee two paths that can be simultaneously pursued given the work done in this thesis: a theoretical path and an experimental path. Future work on the theoretical side could include diversifying the model of the server state to incorporate different types of tasks. Such a scenario could represent a setting where UAV operators have to juggle both the incoming classification tasks, and more trivial tasks such as health monitoring and communication with headquarters. Handling of classification tasks may be very intensive for the server, while the remaining tasks could be less demanding and may even help to lower the server's state, for example if the server receives positive feedback on his performance from headquarters. An interesting area to investigate would hence be the ordering of intensive and non-intensive tasks such that the server's performance is optimal.

A second area we are interested in is the development of a decision-aid that could assist the server in making better decisions on how to allocate his attention. For example, given the server model, policies could be developed to inform the server of how much time he should spend on a task and to prompt him to move on when the time is up. On the other hand, such policies would also be able to rely on data gathered during the experiment to determine how well the server is performing and suggest fixes. For example, if a server spends too little time on a task, then the task controller could reschedule the task for a relook automatically since the probability of the server getting it wrong is high. We foresee such policies to be used in complement with the single-task threshold policy.

Further work that can be done experimentally include investigation of better methods to estimate the server parameters, τ and x_0, and methods to derive the server's self-determined weightage placed on either getting a question right or getting it done quickly. Possible methods that have been briefly looked at for the former case include Hidden Markov Models or the EM method, while the latter case might require more feedback from the server through qualitative questions asked throughout the experiment.

Appendix A

Experiment Documents

Effect of task admission control on productivity of humans

Experiments for Dec. 2010

Christine Siew

MIT ARES LIDS

10 December 2010

The Experiment

- 45-minute long experiment
- Answer as many analogy questions as possible, with no possibility of skipping questions
- Your participation will aid us in the development of task allocation strategies for increased productivity
- A short survey will be given at the beginning and at the end of the experiment for feedback
- Give yourself a unique ID which will be used to label your pre- and post- experiment surveys.
- For more information, please contact: **Christine Siew (chrisiew@mit.edu)**

Your Objective:

- Answer as many questions as correctly as you can **AND** as fast as possible within allocated time frame
- You will be awarded 1 point for every correct answer
- Incorrect answers will be penalized with a subtraction of 1 points from your total score per incorrect answer
- Maximize your score by:
    - Maximizing number of correct answers
    - Minimizing incorrect answers
    - Answering the questions quickly

Introduction to Analogy Questions

- Analogy questions test your ability to recognize the relationship between the words in a word pair and to recognize when two word pairs display parallel relationships
- To answer an analogy question, first formulate a relationship between the words in the question word pair. Next, select the word pair from the options given that are related to one another in most nearly the same way

Example Analogy Question

WALK : LEGS

A. blink : eyes

B. chew : mouth

C. dress : hem

D. cover : book

E. grind : nose

Example Analogy Question

WALK : LEGS

A. blink : eyes

B. chew : mouth

C. dress : hem

D. cover : book

E. grind : nose

Participant Identification Number:

____

1. **Please Indicate your Sex: Male / Female**

2. **Please Indicate your Age:** ____________________

3. **Please Indicate your Occupation (If Student, Indicate your Year and Degree)**

4. **Please Indicate your Primary Spoken Language**

5. **Rate your Command of English on a Scale of 1 – 5**
   (1:poor and 5:excellent)

   1 2 3 4 5

6. **Indicate the Number of Years you have been Studying English** _________________

7. **Have you ever Taken the Verbal SAT? Yes / No**

8. **Degree of Familiarity in Solving Verbal Analogy Questions**
   (1: Unfamiliar and 5: Very familiar)

   1 2 3 4 5

**CONSENT TO PARTICIPATE IN
NON-BIOMEDICAL RESEARCH**

EFFECT OF TASK ADMISSION CONTROL ON PRODUCTIVITY OF HUMANS

You are asked to participate in a research study conducted by Professor Emilio Frazzoli, Ph.D., from the Aeronautics and Astronautics Department at the Massachusetts Institute of Technology (M.I.T.). You were selected as a possible participant in this study because you have a good command of the English language and good eyesight with no color blindness, which is important for participation in this experiment. The population that this research will eventually influence would be human operators required to engage in persistent tasks. You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate.

- **PARTICIPATION AND WITHDRAWAL**

Your participation in this study is completely voluntary and you are free to choose whether to be in it or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise which warrant doing so.

- **PURPOSE OF THE STUDY**

The objective of this experiment is to investigate the effects of task admission control on the productivity of humans.

- **PROCEDURES**

If you volunteer to participate in this study, we would ask you to do the following things:

- Participate in a timed multiple-choice GRE-like Analogy test or/
- Participate in a timed test involving identifying the number of features (e.g. airplanes, cars) from static pictures

Following the test, you would be asked to do the following things:

- Fill out a feedback survey.
- Total time: Approximately 1 hour.

- **POTENTIAL RISKS AND DISCOMFORTS**

There are no anticipated physical or psychological risks in this study.

- **POTENTIAL BENEFITS**

While there is no immediate foreseeable benefit to you as a participant in this study, your efforts will provide critical insight into the development of a methodology that can help researchers develop guidelines for support system design for human operators.

- **PAYMENT FOR PARTICIPATION**

You will be paid $10 to participate in this study. This will be paid upon completion of your debrief and/or through the distribution of gift certificates to your primary email address upon completion of the study[1]. Should you elect to withdraw in the middle of the study, you will be compensated for the time you spent in the study.

- **CONFIDENTIALITY**

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law.

You will be assigned a subject number which will be used on all related documents to include databases, summaries of results, etc. Only one master list of subject names and numbers will exist that will remain only in the custody of Professor Frazzoli.

- **IDENTIFICATION OF INVESTIGATORS**

If you have any questions or concerns about the research, please feel free to contact the Principal Investigator, Emilio Frazzoli, at (617) 253-1991, e-mail, frazzoli@mit.edu, and his address is 77 Massachusetts Avenue, Room 33-332, Cambridge, MA 02139. The Research Scientist is Ketan Savla at (617) 324-0095, email, ksavla@mit.edu and the Research Assistant is Christine Siew at (734) 709-6198, email, chrisiew@mit.edu.

[1] Please allow several days for processing.

- **EMERGENCY CARE AND COMPENSATION FOR INJURY**

If you feel you have suffered an injury, which may include emotional trauma, as a result of participating in this study, please contact the person in charge of the study as soon as possible.

In the event you suffer such an injury, M.I.T. may provide itself, or arrange for the provision of, emergency transport or medical treatment, including emergency treatment and follow-up care, as needed, or reimbursement for such medical services. M.I.T. does not provide any other form of compensation for injury. In any case, neither the offer to provide medical assistance, nor the actual provision of medical services shall be considered an admission of fault or acceptance of liability. Questions regarding this policy may be directed to MIT's Insurance Office, (617) 253-2823. Your insurance carrier may be billed for the cost of emergency transport or medical treatment, if such services are determined not to be directly related to your participation in this study.

- **RIGHTS OF RESEARCH SUBJECTS**

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E25-143B, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253 6787.

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

____

Name of Subject

____

Name of Legal Representative (if applicable)

____	______________
Signature of Subject or Legal Representative	Date

SIGNATURE OF INVESTIGATOR

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

____	______________
Signature of Investigator	Date

Participant Identification Number:

Feedback Survey

____

1. **On a scale of 1 to 5 (with 5 being very intense), how intense was the work allocated during the experiment? Did the intensiveness of the work possibly hinder your best performance?**

2. **Was the objective (maximize the number of correct answers, minimize the number of wrong answers and the average time taken to answer a question) clear enough? Do you think you were indeed maximizing this objective? If not, how could the objective be changed?**

3. **What percentage of the questions do you think you got correct? Please explain how you could have achieved a better score.**

4. **What was the strategy you utilized during the experiment to achieve the best possible score?** E.g. Did you allocate a fixed amount of time for each question and guess the answer after that, or did you try to pick the right answer all the time, regardless of the amount of time spent

Bibliography

[1] Federgruen A. and So K.C. Optimality of threshold policies in single-server queueing systems with server vacations. *Advances in Applied Probability*, 23:388–405, 1991.

[2] Wickelgren W. A. Speed-accuracy tradeoff and information processing dynamics. *Acta Psychologica*, 41(1):67–85, 1977.

[3] Donmez B., Nehme C., Cummings M., and de Jong P. Modeling situational awareness in a multiple unmanned vehicle supervisory control discrete event simulation. *Journal of Cognitive Engineering and Decision Making*, 2008. Special Issue on Computational Models of Macrocognition (in review).

[4] Greenshields B.D. The photographic method of studying traffic behavior. In *Proceedings of the 13th Annual Meeting of the Highway Research Board*, 1933.

[5] Kerner B.S. *Introduction to Modern Traffic Flow Theory and Control.* Springer, 2009.

[6] Wickens C.D. *Engineering Psychology and Human Performance.* Harper, 1992.

[7] Blumentritt C.W., Pinnell C., McCasland W.R., Ross D.W., and Glazer J. Guidelines for selection of ramp control systems. Technical report, Transportation Research Board, Washington DC. NCHRP Report 232.

[8] Kendall D.G. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markob chain. *Ann. Math. Statistics*, 24:338–354, 1953.

[9] Heyman D.P. The t-policy for the m/g/1 queue. *Management Science*, 23:775–778, 1977.

[10] Masher D.P., Ross D.W., Wong P.J., Tuan, Zeidler P.L., and Peracek S. Guidelines for design and operating of ramp control systems. Technical report, Stanford Research Institute Report. NCHRP Report 3-22.

[11] Plossl G.W. Throughput time control. *International Journal of Production Research*, 26(3):493–499, 1988.

[12] Wiendahl H.P. *Load-oriented Manufacturing Control.* Springer-Verlag, 1995.

[13] Wiendahl H.P., Glassner J., and Petermann D. Application of load-oriented manufacturing control in industry. *Production Planning and Control*, 3(2):118–129, 1992.

[14] Romani J. Un modelo de la teoria de colas con numero variable de canales. *Trabajos de Estadistica*, 8:175–189, 1957.

[15] Talman A. J. J. A simple proof of the optimality of the best n-policy in the m/g/1 queueing control problem with removable server. *Statistica Neerlandica*, 33:143–150, 1979.

[16] Harrison J.M, Holloway C.A., and Patell J.M. Measuring delivery performance: case study from the semi-conductor industry, paper presented at the 'measuring manufacturing performance' colloquium. Harvard Business School, January 1989.

[17] Bertrand J.W. and Wortmann J.C. *Production Control and Information SYstems for Compenent Manufacturing Shops.* Elsevier, 1981.

[18] Cohen J.W. and Boxma O.J. A survey of the evolution of queueing theory. *Statist. Neerlandica*, 39:143–158, 1985.

[19] Savla K., Nehme C., Temple T., and Frazzoli E. On efficient cooperative strategies between uavs and humans in a dynamic environment. In *AIAA Conference on Guidance, Navigation and Control*, 8 2008. Electronic Proceedings.

[20] Boettcher K.L. and Tenney R.R. Distributed decision-making with constrained decision makers: A case study. *IEEE Transactions on Systems, Man and Cybernetics*, 16(6):813–823, 1986.

[21] Balachandran K.R. Control policies for a single server system. *Management Science*, 19:1013–1018, 1973.

[22] Balachandran K.R. and Tijms H. On the d-policy for the m/g/1 queue. *Management Science*, 21:1073–1076, 1975.

[23] Kleinrock L. *Queueing Systems*. Wiley, 1975.

[24] Tadj L. and Choudhury G. Optimal design and control of queues. *Sociedad de Estadistica e Investigacion Operativa*, 13(2):359–412, 2005.

[25] Cummings M. and Mitchell P. Predicting controller capacity in remote supervision of multiple unmanned vehicles. *IEEE Systems, Man and Cybernetics*, 38(2):451–460, 2008.

[26] Papageorgiou M. and Kotsialos A. Freeway ramp metering: An overview. In *IEEE Intelligent Transportation Systems*, 2000.

[27] Yadin M. and Naor P. Queueing systems with a removable service station. *Operational Research Quarterly*, 14:393–405, 1963.

[28] Yadin M. and Naor P. On queueing systems with variable service capacities. *Naval Research Logistics Quarterly*, 14:43–54, 1967.

[29] Farhi N., Goursat M., and Quadrat J.P. Fundamental traffic diagram of elementary road networks. In *ECC*, 2006.

[30] Bekker R. Finite buffer queues with workload-dependent service and arrival rates. *Queueing Systems*, 50:231–253, 2005.

[31] Bekker R. *Queues with State-dependent Rates.* PhD thesis, Eindhoven University of Technology, 2005.

[32] Bekker R. and Borst S.C. Optimal admission control in queues with workload-dependent service rates. *CWI*, 2005.

[33] Deb R. Optimal control of batch service queues with switching costs. *Advances in Applied Probability*, 8:177–194, 1976.

[34] Ollman R. Fast guess in choice reaction time. *Psychonomic Science*, 6:155–156, 1966.

[35] Pew R.W. The speed-accuracy operating characteristic. *Acta Psychologica*, 30:16–26, 1969.

[36] Srivastava V., Carli R., Bullo F., and Langbort C. Task release control for decision making queues. In *American Control Conference*, 2010. Submitted.

[37] Bechte W. Theory and practice of load-oriented manufacturing control. *International Journal of Production Research*, 26(3):375–395, 1988.