Energy-Aware Multiserver Queueing Systems with Setup Times: Structural Properties, Exact Analysis, and Asymptotic Performance

I dedicate this thesis to my wife Stephanie. It would not have been possible without her constant support.

Abstract

Energy consumption of today's data-centers is a constant concern from the standpoints of monetary and environmental costs. An intuitive solution to address these immense energy demands is to turn servers off to incur less costs. As such, many different authors have modeled this problem as an $M/M/C$ queue where each server can be turned on, with an exponentially distributed setup time, or turned off instantaneously. What policy the model should employ, or rather when each server should be turned on and off is far from a trivial question. A specific policy is often examined, but determining which policy to study can be a difficult process and is often a product of intuition. Moreover, while a specific policy may do comparatively well against another, in general it may be far from optimal. This problem is further accentuated when one considers the case that a policy may do well or even be optimal under a specific cost function, but far from optimal under another. To address this issue this thesis studies the structural properties of the optimal policy under a wide range of cost functions, allowing for a significant reduction in the search space; then leverages these structural properties to intelligently select three families of policies to study further. An exact analysis of these policies is given, alongside offering insights, observations, and implications for how these systems behave. In particular, results are found which grant insight into the question of the number of servers that should remain on at all times under a general cost function. The model is then analysed under a fixed-load, many-server asymptotic regime, i.e. $C \to \infty$, while the load remains fixed, i.e. $0 < \rho < 1$, where it is shown that not only are many of the policies in the literature equivalent under this regime, but they are also optimal under any cost function which is non-decreasing in the expected energy cost and response time.

Acknowledgements

Firstly, and most importantly, I must express my sincerest gratitude to Dr. Douglas Down, my supervisor of over five years. His profound guidance through the academic landscape was invaluable in the formation and completion of this thesis, as well as all other aspects of my academic career. His keen and vigilant editorial skills were an incredible asset throughout. And his willingness to help and good nature were paramount to my overwhelming positive experiences a graduate student. I cannot thank him enough.

Secondly, I would like to acknowledge the guidance of my advisory committee. Their constructive feedback and willingness to help steer my research is greatly appreciated.

Lastly, this research was funded by the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Figures

Chapter 1

Introduction

The immense energy consumption of data-centers has become a fact of modern life. The United States spends on the order of billions of dollars powering these systems each year [12, 37], and a large standalone corporation such as Google pays an annual energy bill on the order of hundreds of millions of dollars [52, 60]. While some may see this as an obligatory cost, the truth is many of these servers spend a significant amount of time idle. Moreover, an idling server uses a large percentage of the energy it would if it were busy [5]. To conserve costs, servers often have a lower energy state they can be switched to (off, sleep, etc.). However, the choice of if and when to make such a switch for each server is far from trivial.

Due to these extravagant costs and open questions regarding data-centre behaviour, a great amount of interest has been cultivated. Specifically, can something be done to avoid said costs while at the same time achieving reasonable performance? Due to the complexity of these systems, direct analysis is intractable, and researchers fall back on two broad strategies, the first being simulation and the second being modelling.

With no discredit to the merits of simulation, this thesis focuses strictly on modelling a data-centre and directly analysing said model.

At its simplest description, the model is one where jobs arrive, are served/processed, and then depart. To make the model interesting (and more true to life), time between job arrivals, alongside job sizes (how long it takes to serve/process a job) follow some statistical distribution; a considerable amount of uncertainty is present. Moreover, since data-centres have many machines, the model allows for many servers. To capture how one can potentially save energy, each of these servers can independently be turned on (after a setup delay which also follows a statistical distribution) or off (instantly). The idea is that one has the ability to turn a server off to save operating costs such as energy, but does so with the penalty of potentially harming system performance. If a server is off, jobs are more likely to experience longer wait times in queue, and in turn a larger response time.

At a glance, one may determine there to be a direct trade-off between efficiency (energy costs) and efficacy (system performance). However, things are more complicated than they initially seem. For example, when one turns a server off, they risk incurring more costs in the long run due to that server now having to expend energy to turn back on, as well as risk system *thrashing* (servers frequently turning on and off without processing many jobs). Therefore, it is not safe to assume turning a server off will decrease costs. This is an unfortunate observation since one can not obviously determine *if* energy costs can be saved, let alone *how*. Complexities such as these are what make the model appealing to study. Moreover, as will be seen, the insights one

can draw from the model are often practical and counterintuitive.

As stated previously, the main interest lies in determining the *best* way to manage the system/model (when to turn servers on and off) with regards to cost and performance. A full description of how to manage the system is referred to as a policy. To determine which policy is the best (or more formally, optimal) there must be a way to compare them. This comparison is done via a cost function which is dependent on appropriate system metrics. That is, letting $\mathcal{C}_\pi$ denote the value of the cost function under policy π, given two policies π_1 and π_2, π_1 is considered a better policy if and only if $\mathcal{C}_{\pi_1} < \mathcal{C}_{\pi_2}$. Moreover, π^* is an optimal policy if and only if for all π, $\mathcal{C}_{\pi^*} \leq \mathcal{C}_\pi$.

With these notions in mind an informal and brief overview of the contributions of this thesis can be made explicit. That is, the contributions of this thesis include but are not limited to:

1. The description of a formal model, referred to as an energy-aware system, and two corresponding Markov decision processes which can be used to determine an optimal policy for said model.

2. Formal proofs regarding several novel, insightful, and counterintuitive structural properties which the optimal policy is guaranteed to adhere to under a wide range of cost functions.

3. A formal description of the optimal policy via threshold decision variables, alongside the formal description of three different sets of policies which approximate the optimal policy while considerably reducing the search space.

4. A large suite of numerical results of the three aforementioned policy sets, showing the existence of configurations where one can get reasonably close to optimal with regards to efficiency and efficacy simultaneously.

5. Several key observations regarding the behaviour of these policies including the existence of a simple static provisioning which does reasonably well most of the time.

6. A formal proof showing that as the number of servers approaches infinity while the load on the system remains fixed, a large set of policies all become optimal under a large set of cost functions.

7. A suite of numerical experiments showing the convergence rates of the previous optimality result, alongside observations to induce this optimal behaviour more quickly.

The organization of this thesis is as follows:

- **Chapter 2 *Preliminaries:*** A brief overview regarding the technical material needed, alongside several examples based on simplified versions of the model employed throughout. If one is confident in their understanding of stochastic modelling and Markovian mechanisms, this chapter could be glossed over or skipped without a fundamental disconnect from the rest of the content.

- **Chapter 3 *Literature Review:*** An overview of the great number of methods and approaches used by researchers to tackle the problem of energy provisioning in data-centres. Moreover, the chapter gives a finer-grained overview of theoretical works examining the same (or similar) models examined in this work, to

illuminate the specific contributions of this thesis, as well as where this work fits in the literature overall.

- **Chapter 4** ***Problem Formulation and Approach:*** This chapter formally defines the model described previously in the introduction. Likewise, a formal definition of the other components which make up the problem are also defined; namely, policies and cost functions. With the model, policy, and cost function given, the chapter gives an overview to the approach and methodology which the thesis employs, and gives the reader a better appreciation for the significance of the contributions.

- **Chapter 5** ***Structural Properties:*** The first of the *three-pronged approach* used in this thesis, this chapter examines the structure of the optimal policy. Specifically, it gives formal results which, if not adhered to, imply certainty of sub-optimality. Moreover, several of these properties are counterintuitive, further adding to their value.

- **Chapter 6** ***Exact Analysis:*** Leveraging the structural properties of the previous chapter, three formal, but fundamentally different, families of policies are defined. An exact analysis is performed on their underlying continuous time Markov chains, from which a large suite of numerical experiments is given. From these experiments several key observations are made and discussed regarding the system behaviour.

- **Chapter 7** ***Asymptotic Performance:*** Policies of the model are further analysed under a fixed-load, many-server asymptotic regime. Under this regime it is shown that a large set of policies are equivalent and optimal under a large

set of cost functions.

- **Chapter 8** ***Conclusion:*** This thesis concludes and provides a discussion of the potential extensions and future work for this research.

- **Appendices** ***Proofs:*** All proofs of the results given in Chapters 5 and 7 are shown in detail, alongside the proof of a lemma which after finalization of this work, was left unused in the main results.

This thesis is a comprehensive and augmented compilation of the work given in [41, 42, 43, 45, 47]. It may also be viewed as an extension of the work [40, 44, 46].

Chapter 2

Preliminaries

This chapter is included for the sake of completeness and clarity. If the reader feels they are comfortable with the concepts of continuous time Markov chains (CTMCs), $M/M/C$ queues, and/or Markov decision processes (MDPs) it is encouraged that they skip or gloss over this chapter. If the reader has further interest in the material presented here, they are directed to [27, 28, 36] for a much more detailed overview of stochastic modelling.

2.1 Stochastic Process

A stochastic process is a mathematical abstraction used to represent and model a randomly changing system over time. Formally, a stochastic process is a set of random variables (or a set of vectors of random variables) $\{X_t \mid t \in T\}$, where the index set T is typically interpreted as a set of time values, and the random variables X_t denote state information of the system in question and may be either discrete or continuous. For example, at time t, X_t may be the number of customers, the energy consumed,

remaining service time of the current customer, etc. In contrast, T may be countable, i.e. $T = \{1, 2, 3...\}$ or defined on some interval, i.e. $T = \mathbb{R}^+$. Informally, one can view a stochastic process as a time dependent random variable.

Specializations of these stochastic processes can be instantiated under particular assumptions. A stochastic process is said to be a Markov process if the Markov property holds. The Markov property states that for every sequence of increasing time values $(t_0, t_1, t_2, ..., t_n)$, given the values of $X_{t_0}, X_{t_1}, X_{t_2}, ..., X_{t_{n-1}}$, the conditional distribution of X_{t_n} depends only on $X_{t_{n-1}}$. This is seen formally as,

$$\begin{aligned} &P[X_{t_n} \leq x_n \mid X_{t_0} = x_0, X_{t_1} = x_1, X_{t_2} = x_2, ..., X_{t_{n-1}} = x_{n-1}] \\ &\quad = P[X_{t_n} \leq x_n | X_{t_{n-1}} = x_{n-1}]. \end{aligned}$$

This has the interpretation that the conditional distribution of the current value depends only on the most recent value known. Taking the most recent known value as the *present* time, one can say that the future values depend only on the present value, and are completely independent from past values. This is often referred to as the process being *memoryless*. Exploiting the memoryless property of a Markov process often allows for an elegant analysis of such systems, since one can analyse all future behaviours of a system with only the knowledge of the current system state. This result is the cornerstone for the analysis of many stochastic models.

2.1.1 Continuous Time Markov Chains

A continuous time Markov chain (CTMC) is a Markov process where the random variables $\{X_t \mid t \in T\}$ take on discrete values from some set S, called the state space, and the set T is defined on some continuous interval. A CTMC is often thought of as a directed graph where the vertices of the graph are the elements of S (the system states), and the edges are the *transition rates* between states, labelled by $q_{i,j}$, denoting the rate at which the system moves from state i to state j. Given the transition rates, one can construct the transition matrix for a given Markov chain as shown in (2.1),

$$Q = \begin{pmatrix} -q_{0,0} & q_{0,1} & q_{0,2} & \cdots \\ q_{1,0} & -q_{1,1} & q_{1,2} & \cdots \\ q_{2,0} & q_{2,1} & -q_{2,2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{2.1}$$

where $q_{i,i} = \sum_{j \neq i} q_{i,j}$. This last relation comes from the fact that for each state, the sum of probabilities to move to any other state (including the given state) equals 1. Table 2.1 shows the different classes of stochastic processes when switching between discrete and continuous state spaces.

	Time Values	
State Space	Discrete	Continuous
Discrete	Discrete-Time Markov Chain	Continuous-Time Markov Chain
Continuous	Discrete-Time Markov Process	Continuous-Time Markov Process

Table 2.1: Stochastic Process Classes

One important special case of CTMCs is the *birth-death process*. Here, there exists a mapping from the state space of the CTMC to a subset of the natural numbers, i.e.

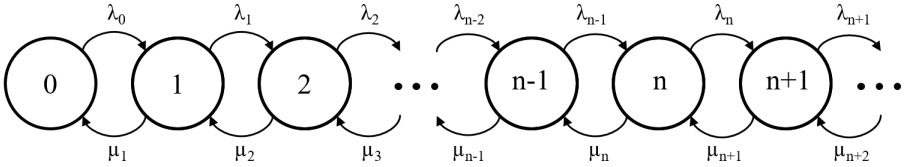


Figure 2.1: General birth-death process

$S \subseteq \mathbb{N}$. For example, the state space may be the number of jobs in the system, the number of servers currently online, etc. Moreover, transitions between these states can only be made in a step-wise manner. That is, a single event can only increase/decrease the current state by a value of one. Letting λ_i denote the rate at which the state space is increased by one in state i (thought of as the birth rate in state i), and letting μ_i denote the rate at which the state space is decreased by one in state i (thought of as the death rate in state i), one can define a CTMC via its transition matrix:

$$Q = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & \cdots \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 & \cdots \\ 0 & \mu_2 & -(\lambda_2 + \mu_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

This can also be seen graphically in Figure 2.1. The general solution to CTMCs of this form is well understood. Letting π_n denote the steady-state probability of being in state n, i.e. $\pi_n = \lim_{t\to\infty} P(X_t = n)$,

$$\pi_n = \pi_0 \frac{\Pi_{i=0}^{n-1} \lambda_i}{\Pi_{i=1}^{n} \mu_i}, \qquad \sum_{i=0}^{\infty} \pi_i = 1.$$

Furthermore, if the state space denotes the number of jobs in the system, the expected

number of jobs, $\mathbb{E}[N]$, follows immediately as

$$\mathbb{E}[N] = \sum_{i=1}^{\infty} i\pi_i.$$

From here, there are two well known results which are used throughout this thesis, and aid with further analysis of these systems. The first is Little's law, which states that the long term mean number of customers or jobs in a stable system is equal to the product of the arrival rate to said system and the mean time a job spends in said system, where the time a job spends in the system is referred to as the response time. Letting λ denote the arrival rate and $\mathbb{E}[R]$ denote the expected response time, this result is more commonly stated as,

$$\mathbb{E}[N] = \lambda\mathbb{E}[R].$$

This is a powerful result as it allows one to relate the expected number of jobs in system to the expected response time, which can be notoriously difficult to arrive at directly. Moreover, Little's law holds with incredible generality; it is independent of the arrival distribution, the number of servers, routing, the employed service policy, etc., making it one of, if not the most important result, in stochastic modelling.

Looking back to the general birth death process, applying Little's law allows one to express the expected response time via,

$$\lambda = \sum_{i=0}^{\infty} \lambda_i\pi_i \qquad\qquad \mathbb{E}[N] = \lambda\mathbb{E}[R].$$

The second result, although not as widely used as Little's law, is important to understanding the analytic techniques used in this thesis. This is the renewal reward theorem, which says that given a *renewal process* (here a renewal interval is taken to be the time between leaving a reference state and returning to it), the expected amount of a *reward* gained, divided by the expected length of the renewal interval equals the expected rate of said reward gained by the entire system. As an example, take the general birth death process from Figure 2.1 and let the renewal interval be the time from the system leaving state 0 (when it is empty) until it returns to state 0, and let the reward be the energy consumed by the system. Letting E_0, T_0, and E denote the total energy consumed during the renewal interval, the length of the renewal interval, and the rate of energy consumption of the system, respectively, the renewal reward theorem states that

$$\mathbb{E}[E] = \frac{\mathbb{E}[E_0]}{\mathbb{E}[T_0]}.$$

This turns out to be extremely convenient, as it allows for metrics which are typically difficult to determine directly, to be expressed by (what is often the case) relatively easy values to obtain.

2.2 The M/M/C Queue

As an example of the typical methodology for these CTMCs, as well as their applications, an analysis of the well understood $M/M/C$ queue is presented. An $M/M/C$ queue is a system which meets the following criteria.

1. Arrivals follow a Poisson process with rate λ. This is equivalent to stating times

between arrivals are exponentially distributed with rate λ.

2. Jobs arrive to a central queue and are served on a first come first serve (first in first out) basis.

3. There are C homogeneous servers. An idle server will take the job at the front of the queue and begin processing it. Job service times are exponentially distributed with rate μ and servers only process one job at a time.

The notation $M/M/C$ follows Kendall's notation. The first M is a shorthand for Markovian arrivals, the second M is a shorthand for Markovian service times, and the C is a shorthand for the number of servers. Letting the state space be the number of jobs in the system (in queue and in service), an $M/M/C$ queue can be modelled as a birth-death process where:

$$\forall i : \lambda_i = \lambda, \text{ and } \mu_i = \min(i, C)\mu.$$

The corresponding CTMC is illustrated in Figure 2.2. Applying the birth-death formulas, after some algebra it can be shown that

$$\pi_i = \begin{cases} \pi_0 \frac{(C\rho)^i}{i!}, & \text{if } 0 < i < C \\ \pi_0 \frac{C^C \rho^i}{C!}, & \text{if } C \leq i \end{cases} \qquad \text{and} \qquad \pi_0 = \left[\left(\sum_{i=0}^{C-1} \frac{(C\rho)^i}{i!} \right) + \frac{(C\rho)^C}{C!} \frac{1}{1-\rho} \right]^{-1}$$

where $\rho = \lambda/(C\mu)$. Furthermore, for stability, it is assumed $\rho < 1$. Continuing with the approach described previously for birth-death-processes, it then follows that

$$\mathbb{E}[N] = \frac{\lambda}{\mu} + \pi_0 \frac{\rho(C\rho)^C}{(1-\rho)^2 C!},$$

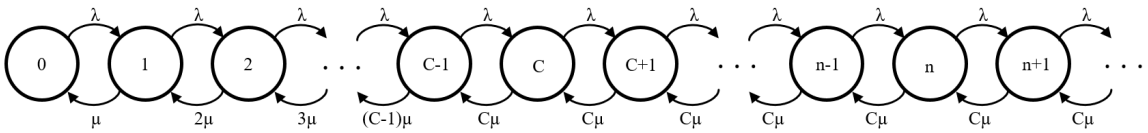


Figure 2.2: CTMC of an $M/M/C$ queue

and after applying Little's law

$$\mathbb{E}[R] = \frac{1}{\mu} + \pi_0 \frac{\rho(C\rho)^C}{\lambda(1-\rho)^2 C!}.$$

2.3 Markov Decision Process

To demonstrate the framework of a Markov decision process (MDP) consider an $M/M/C$ queue with a small alteration. Control is imposed such that the queue is not forced to serve a job when possible, but cannot stop processing a job once it has started. That is, the system (or manager of that system) can choose to leave a server idle rather than serve a job from the queue. For this decision to be non-trivial, there must be some incentive to process a job, as well as let a server idle.

Before these incentives are made clear, some notation must first be introduced. Let $\mathcal{S}$ denote the set of states in the MDP. Here $\mathcal{S} = \{(i,j) \mid j \in \mathbb{N}, 0 \leq i \leq \min(j, C)\}$, where j denotes the number of jobs in the system, and i denotes the number of servers busy processing jobs. Let $\mathcal{A}$ denote the set of actions the control can make. Here an action is equivalent to how many servers will be busy until the next decision is made. Therefore, $\mathcal{A} = \{a \mid 0 \leq a \leq C\}$. However, it should be clear that in certain states some actions are impossible. For example in state $(2, 0)$ the control cannot choose for

five servers to be busy, as there are not enough jobs present to do so. Similarly, in state $(4, 4)$ the control cannot choose for only two servers to be busy as the system does not allow service times to be interrupted. As such, $\mathcal{A}_s \subseteq \mathcal{A}$ is defined where $\mathcal{A}_s$ denotes the set of admissible actions in state s. Here, $\mathcal{A}_{i,j} = \{a \mid i \leq a \leq \min(C, j)\}$.

Actions are performed at specific moments in time referred to as *decision epochs*. To determine when these decision epochs occur the MDP often undergoes uniformization, see [59] for further explanation. The main points to be aware of are that after an MDP has been uniformized, the expected times between all decision epochs are equal, and whenever an event occurs (in the example an event is a job departing or arriving), there will also be a decision epoch. Note however the converse does not hold, i.e. it may not be the case that at every decision epoch an event occurs simultaneously (these are often referred to as dummy events). Another consequence of uniformization is that the rates can then be used as probabilities (with the appropriate dropping of units) when determining the next state of the system.

When an action is performed the MDP gains a reward (or incurs a cost), defined by a reward function $w(s, a)$, which denotes the expected reward gained when action a is performed while in state s. By definition it must hold that $a \in \mathcal{A}_s$. An example reward function for the controllable $M/M/C$ queue would be:

$$w((i, j), a) = j + \beta a,$$

where βa captures the cost of running servers, while j captures the cost of having jobs in the system (this is often referred to as the holding cost). The goal is to choose

an action a at each decision epoch such that the sum of rewards (costs) is maximized (minimized) until some time horizon is reached; this time horizon could be infinite. The difficulty lies in the fact that a chosen action may be appealing in the short run, but bring the system to an unfavourable state in the long run. To formalize this notion, more notation must be introduced. Let $p(s'|s,a)$ denote the probability of being in state s' at the next decision epoch, given that at the current decision epoch the system was in state s and performed action a. For the running example,

$$p((i',j')|(i,j),a) = \begin{cases} a\mu & \text{if} \quad i' = a,\ j' = j-1 \\ \lambda & \text{if} \quad i' = a,\ j' = j+1 \\ 1 - a\mu - \lambda & \text{if} \quad j' = a,\ j' = j \end{cases}$$

The MDP is now fully defined. To determine the optimal policy (the optimal action a corresponding to each state), there are two popular algorithms, value iteration, and policy iteration. Neither of these methods are directly employed in this thesis however, and therefore, further interest on the matter is left to the responsibility of the reader, see [59].

Chapter 3

Literature Review

In general, the topic of green computing is an active and thriving area of research. This is partly due to the immense energy costs of modern systems which in turn gives a major incentive to study this area. This is also partly due to the broad nature of the term *green computing*, and the variety of approaches one can take to tackle the problem. This chapter gives reference and summary to a great number of these works, but focuses on research using methodologies and/or tools directly or tangentially related to queueing theory. For approaches to the problem not grounded in queueing theory, the reader is directed to [2, 6, 7, 49, 53, 68].

Chen et al. [9] were among the first (if not the first) to address the issue of turning servers off to conserve energy by defining a formal model. They approached the problem by inspecting the model at discrete time events, and determined the optimal number of servers to switch on/off based on system parameters at said time (workload, current provisioning, etc.). As such, the model lends itself to be described as an optimization problem if all information was known ahead of time (the optimal policy

could be determined offline). The parameters for the objective function and constraints are determined via known queueing theory formulas. For the online approach they also offered a control theory methodology. For both approaches however, due to setups happening instantaneously, the discrete time units (time between decision epochs) had to be large enough such that actual setup times were negligible. The end result is that their model is geared towards looking at these systems over larger time scales, where some of the finer grained details may be lost.

A similar optimization problem was examined by Liu et al. [39]. Here, they again determined the optimal number of servers to be operating at a decision epoch, but over a much larger (geographical) scale. That is, they account for work loads arriving from all over the country (United States of America) and which data-centre(s) should said work be routed to under the considerations of latency delays, and energy cost of a specific data-centre (for example different states have different energy costs, taxes, etc.). Moreover, not only are the financial costs considered, but also the environmental cost of a particular choice. For example, one data-centre may be run on fossil fuels, another nuclear, another wind, any combination thereof, and so on. Again however, due to the assumed instant setups, the time scale must be considered to be relatively large to be applicable.

To address the assumption on setup times (or disregard thereof), a considerable number of researchers began modelling data-centres as queueing systems, where a server can be switched on (after a setup delay), or off (sometimes after a delay but more usually instantly). It is worth noting that similar queueing models have existed for

decades, known as *vacation* models [64]. However, there exist subtle differences which inhibit vacation models from being interpreted as an energy-aware system (or individual server). Specifically, in vacation models [4, 13, 30, 55, 57, 63], a server enters its vacation period (interpreted as its setup process in the context of green computing) immediately after switching off; whereas it is likely a server should remain off for an interval of time (or until an event occurs) before beginning to turn on. Therefore, while only slightly different, a new family of models was built up.

Perhaps the most widely recognized and cited of this new family of models is the body of work produced by Gandhi et al. [14, 15, 16, 17, 18, 19, 20, 21]. The work began by examining an $M/M/1$ queue, where the server can be switched on after some setup delay or switched off instantly [15]. Moreover, they studied the model under the *energy-response product* (ERP) cost function, which is the product of the expected response time and the expected energy cost. They showed that under this cost function the optimal policy will either be the one that always keeps the server on, or the one that turns the server off the moment it idles, and begins its setup the moment a job arrives. It is worth noting that the surprising aspect of this result is not that the server will never idle (since it instantly turns off), but rather that it is always optimal to start a server setup as soon as any job arrives, as opposed to waiting for potentially more jobs to accumulate. As the authors note, this is a consequence of the cost function, and is not optimal in general.

Gandhi et al. [19] continued research on similar models but for multiserver models. It is worth noting that in general, multiserver systems are notoriously more difficult

to analyse than the corresponding single server equivalents. They began by looking at the policy which turns servers off the moment they idle; if there is ever a job waiting in the queue while at least one server is off, then exactly one server will be in setup. They were able to give an exact analysis of this policy (but not closed form expressions). Moreover, relaxing the assumption of exponentially distributed service times, they showed decomposition results relating their model to the $M/G/C$ queue via z-transforms. As one may intuitively observe, such a policy has the potential to be incredibly slow to adapt to bursts of traffic (the policy is quick to turn servers off and hesitant to turn servers on). As such in [14], they offered and analysed two other policies. The first policy, instead of turning one server on at a time, keeps the number of servers in setup equal to the number of waiting jobs (when possible), and turns servers off when they idle. The second policy uses the same turn on scheme as the first, but waits an exponentially distributed amount of time before turning an idle server off. As noted by Gandhi et al. [22] and also in this thesis, being liberal with server turn offs can be disastrous. When analysing the underlying CTMCs, Gandhi et al. [14] also introduced a novel approach termed the recursive renewal reward (RRR) method, based around the renewal reward theorem, and an inherent recursion one can exploit within these CTMCs. This thesis also employs this method and its application will be seen in detail in Chapter 6.

Working with the previous research as a foundation (or concurrently to it) other researchers studied the same model. This research can separated into two classes: single server, and multiserver. The review continues firstly, with the single server case. Maccio and Down [40, 44, 46] extended the single server model given in [15] by allowing

for k jobs to accumulate before beginning the setup process, as well as allowing the server to idle for an exponentially distributed amount of time before turning off. This modification allowed the model to describe the optimal policy under a much wider range of cost functions. Moreover, the authors were able to relax the exponential assumption on the service and setup times, and give explicit closed form expressions for the performance metrics. Other authors extended this model further, or used it in a larger framework.

Hyytiä et al. leveraged this model in a number of publications. Specifically, in [33] they analysed the model under the processor sharing (PS) and last come first serve (LCFS) service disciplines (whereas before jobs were processed first come first serve). In [32] they derived closed form expressions for higher moments of the response time, which is used to reason about fairness. In [34] they used the single server model in a multiserver context with routing, expanding on some of the preliminary observations made in [46]. Although the overall system has multiple servers, the model is a single server queue, therefore, it is categorized with other single server models. Gebrehiwot et al. also took this single server model and adapted it. In [23] they added multiple sleep states which the server could power down to (each with its own setup time and energy cost). In [24] they showed that in a multiserver routing scenario (jobs are routed to single server models as before), it can be beneficial to allow servers to idle for a period of time, implying the optimal policy is no longer one which immediately switches a server off. In [25] and [26] they examine the model with batch arrivals under the PS and shortest remaining processing time (SRPT) service disciplines. The idea of batch arrivals became a popular extension as Harrison et al. [29] also studied it, but

relaxed the assumption of instant turn offs (giving the server a power down time in addition to the setup time). In this case, they were able to arrive at exact solutions for the response time and busy time distributions, rather than just the means. Moreover, batch arrivals were also considered by Yajima and Phung-Duc [67] with the addition of speed scaling (the server can process jobs more quickly but at a greater energy cost).

When conducting research on single server models, as may be inferred from the previous summary, the typical approach is to either extend the analysis to provide greater insight, e.g. higher moments of performance metrics, distributions of those performance metrics, routing scenarios, etc., or relax model assumptions to make it more generalized, e.g. general service times, batch arrivals, non-trivial turn offs, etc. Analysing these extensions is possible due to the single server nature of these systems. Specifically, much of the analysis relies on events happening non-concurrently (a single server system cannot be processing a job and be in setup at the same time). This in turn allows for traditional exploitation of Markovian properties, usually by constructing an embedded Markov chain. In the multiserver case however, such extensions are difficult to analyse if one hopes to arrive at explicit expressions using typical methods. In fact, even fundamental queueing problems remain to be solved, e.g. there is no known closed form expression for the expected response time of a classical $M/G/C$ queue. Here difficulties come from the fact multiple servers could be processing multiple jobs/be in setup at the same time. Therefore, the number of parameters required to describe the system state increases, and worse yet, are continuous rather than discrete. This is not to say however, that further research on

multiserver systems with setups is hopeless, in fact far from it, but a different approach is usually taken. Instead of extending the model or relaxing assumptions, one instead studies a typical $M/M/C$ queue with setups under a novel policy or family of policies (when to turn servers on/off). Due to the aforementioned complexities which the multiserver models exhibit, such an examination of a specific family of policies can often times be insightful, and in most cases novel in its own right.

Mitrani [50] studied this model where a reserved set of servers are brought into setup when the number of jobs in the system exceeds a threshold, and then shuts those servers off once the number of jobs drops below another threshold. In addition to this reserve of setup servers, a static set of servers is always available to the system at all times. The author was able to derive the moment generating function of the response time via a z-transform apporach. This policy was further studied in [31]. Xu and Tian [66] studied the set of policies where e servers are turned off when there are d servers idle. Giving an exact analysis of the underlying CTMC allows one to perform numerical experiments (as is often the case). Kuehn and Mashaly [38] analysed policies which wait for a threshold number of jobs to accumulate in the queue before a server starts its setup and turns servers off when they idle, under the presence of a finite buffer. Shortly after this publication, Phung-Duc analysed the same variant (finite buffer and threshold) in [54]. In addition, Phung-Duc [56] gives an alternative analysis to the multiserver policies analysed by Gandhi et al. and in [58] analyses a threshold policy, similar to the one examined in Chapter 6, with the addition of customer abandonments. Lastly, Ren et al. [61] analysed a finite two-dimensional

CTMC similar to Kuehn and Mashaly in the context of virtual networks, which allows for a number of servers to always remain operational, but omits the use of turn on thresholds.

In addition to the directly related work described above, there are a few examples of these models (or similar models) being used in different contexts/domains. For example, a similar formulation of the corresponding MDP is given in [48] but in the context of logistics/shipping. It is also natural to extend these models to the domains of manufacturing, where a server would be a machine which could be switched on and off. This idea is explored in [3]. Perhaps the most closely related context however, is speed scaling (which was briefly mentioned previously in this chapter). Here the server can process jobs at a faster rate but at a higher energy cost [10, 11, 65]. At a glance it seems fundamentally different to the setup models. However, it is the authors' opinion that these models share an underlying commonality. Specifically, if setup times in the setup models were assumed to be instant, or relatively quick, one could approximate or directly mimic the speed scaling case. That is, one could turn more servers on (increase the service rate of the system) at a greater energy cost to imitate the effect of speed scaling. There do exist complications however. Specifically, setup models have linear energy cost (number of servers on/busy), while the speed scaling usually has an energy cost cubic in the service rate. One way around this is to look at the structure of these policies (as done in Chapter 5) under an abstract but monotonically increasing energy cost.

Although a great deal of research has been performed in the area, a great deal of open

problems still exist, specifically, those related to the optimal policy in the multiserver context. Moreover, problems exist with the current methodology when researching these systems. These issues, and how they are addressed, are discussed in the next chapter.

Chapter 4

Problem Formulation and Approach

A high level view of the problem examined in this thesis can be seen as follows. Jobs arrive to an energy-aware system in a non-deterministic fashion, where a manager has control over each server. That is, the manager has the ability to turn servers on to increase efficacy and off to increase efficiency. A full description of when to turn each server on and when to turn each server off is referred to as a policy. Moreover, the manager has an objective or cost function which they wish to optimize. That is, the manager wishes to determine a policy such that their costs are minimized. As such, the problem can be thought of as three abstract components: an energy-aware system, a policy, and a cost function. An abstract graphical representation of how these components interact can be seen in Figure 4.1.

The difficulty of this problem lies in the three required steps shown in Figure 4.1. Firstly, when choosing a policy to analyse further, the set of potential policies one

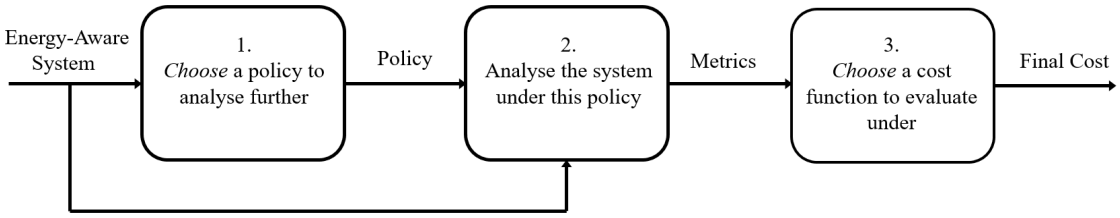


Figure 4.1: A broad overview of this thesis' problem and approach

could choose is infinite, and moreover, the conclusions one draws from said analysis could be extremely sensitive to this choice. Secondly, analysing a specific policy is far from trivial. It often requires a significant amount of time and care to arrive at any conclusions from numerical studies. Lastly, once the analysis is complete and closed form expressions for the system metrics arrived at, one is again faced with choosing a specific element from an infinite set, i.e. one must choose a cost function. But again, the conclusions one draws from a policy, that is, whether that policy is favourable to implement, is sensitive to one's choice of cost function. In other words, a specific policy may be appealing under one cost function, but disastrous under another. Fortunately, this thesis provides counterintuitive results which allows one to overcome most, if not all of these issues. The overview of the problem will be revisited with finer detail once the components of the problem are formally defined.

4.1 Model

An energy-aware system is modeled as an $M/M/C$ queue where each server can be switched on and off, and where turn-offs are instantaneous, but turn-ons take an exponentially distributed setup time. This is described formally as follows. Jobs arrive

to a central infinite queue following a Poisson process with rate λ, are processed on a first come first served basis, and have service times (job sizes) which are exponentially distributed with rate μ. Furthermore, there are C homogeneous servers, each of which can be in one of four energy states: *off, setup, idle*, or *busy*. For ease of exposition this work often refers to a server being busy, idle, off, or in setup as shorthand for a server being in the corresponding energy state. Regarding definitions and transitions, when a server is *off* it may begin turning on by moving to *setup*. Once in *setup* the server will remain there for an exponentially distributed amount of time with rate γ before it is turned on and becomes *idle* or *busy*. When on, the server is *idle* if it is not processing a job and *busy* if it is. Servers are free to move between *idle* and *busy* as long as there is a job present which it could serve. Furthermore, at any time a server can instantly be switched *off*. It is worth noting that this implies a server's setup process can be canceled (moved from *setup* to *off*). The model of study can be viewed graphically in Figure 4.2.

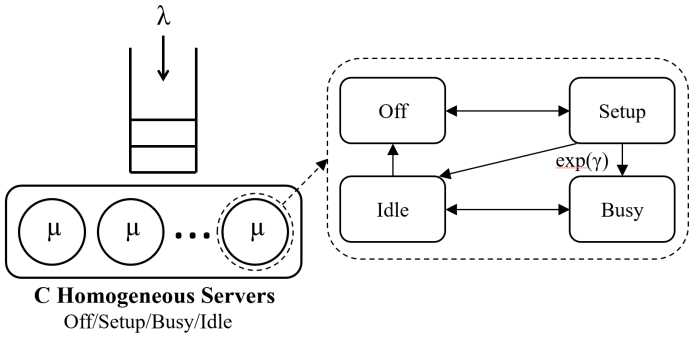


Figure 4.2: The model under study. Dynamic servers take time exponentially distributed with rate γ to move from *setup* to *idle* or *busy*, all other transitions happen instantly if the system state allows it.

Formally an energy-aware system can be viewed as a four-tuple $(C, \lambda, \mu, \gamma)$, where C

is the number of servers, and λ, μ, and γ are the arrival, processing, and setup rates, respectively. The system load ρ is defined as $\rho = \lambda/(C\mu)$. For ease of reference this definition is restated below.

Definition 4.1. ***Energy-Aware System:*** *An energy-aware system is formally defined as a four-tuple $(C, \lambda, \mu, \gamma)$, where C is the number of servers, and λ, μ, and γ are the arrival, processing, and setup rates, respectively.*

These underlying four parameters of an energy-aware system are usually thought to be given, that is to say there does not exist any control to modify these values. One does have control however, of when each of these C servers are turned on/off. As mentioned before, a precise description of this server behaviour is referred to as a policy. For example, given a two server system ($C = 2$), one could describe a policy as: turn the first server on if there are two or more jobs in the system, turn the second server on if there are five or more jobs in the system, turn both servers off when the system becomes empty. Of course as C gets larger describing a policy in this way can become tedious. As such, policies are often described in a broader fashion. Some examples of these descriptions are: the number of servers which are on or in setup equals the number of jobs in the system, turn all servers on once there are k jobs in the system and turn them off when they idle, keep all servers on all the time, etc. Often times when describing these policies a set of servers is provisioned which always remain on. A server which always remains on is referred to as *static*. Furthermore, a server which regularly turns off and on is referred to as *dynamic*. Throughout this work a specific policy is often denoted by π.

Regarding these energy-aware systems, the set of potential policies one can study is

infinite, but to make a recommendation on which policy to implement, this infinite set must be ordered; policies must be able to be compared against each other. To achieve this ordering, one first needs to associate metrics with an energy-aware system employing a given policy. In other words, one must define what it means for a policy to be better than another. This work examines the trade-off between efficacy and efficiency. The expected response time, denoted by $\mathbb{E}[R]$ is employed to evaluate efficacy, while the expected energy cost, denoted by $\mathbb{E}[E]$, is employed to evaluate efficiency. The expected response time is defined as the expected amount of time a job spends in the system, from arrival to departure. Defining the expected energy cost takes a little more care. Each of the energy states (*off, idle, busy*, and *setup*) has a corresponding energy consumption rate. Let these rates be denoted by E_{Off}, E_{Idle}, E_{Busy}, and E_{Setup}, respectively. Furthermore, let the random variables C_{Off}, C_{Idle}, C_{Busy}, and C_{Setup} denote the number of servers which are off, idle, busy, or in setup, respectively. Then

$$\mathbb{E}[E] = E_{\text{Off}}\mathbb{E}[C_{\text{Off}}] + E_{\text{Idle}}\mathbb{E}[C_{\text{Idle}}] + E_{\text{Busy}}\mathbb{E}[C_{\text{Busy}}] + E_{\text{Setup}}\mathbb{E}[C_{\text{Setup}}], \tag{4.2}$$

where it is assumed throughout this work that $E_{\text{Off}} = 0$; this assumption could be relaxed to account for lower energy consumption states where the server cannot process jobs, e.g. sleep states. Moreover, an inherent ordering is imposed on the energy rates. That is, $E_{\text{Off}} < E_{\text{Idle}} < E_{\text{Setup}}$ and $E_{\text{Idle}} < E_{\text{Busy}}$ (note that no assumption is imposed on the ordering of E_{Busy} relative to E_{Setup}). Often times it is advantageous to consider only the expected *excess* energy cost. Here, one is only interested in the cost contributed by servers which are idle or in setup. This is done because, as it will be shown, energy costs due to busy servers is independent from which policy is being

implemented. However, unless it is stated explicitly within the chapter, $\mathbb{E}[E]$ denotes the expected *total* energy cost, and not the expected *excess*.

One last way one can view the expected energy cost is on a per job basis. That is, letting E^J be a random variable denoting the energy cost contributed by some job J, then by definition

$$\mathbb{E}[E^J] = \mathbb{E}[E]/\lambda.$$

This is often useful (as will be seen in Chapter 7) when system parameters are allowed to go to infinity. This is due to the possibility of $\mathbb{E}[E]$ being infinite, while $\mathbb{E}[E^J]$ may remain finite.

From here one can begin to build cost functions from the system metrics. While specific and popular cost functions do arise in the literature, e.g. $\mathbb{E}[R]\mathbb{E}[E]$, and $\mathbb{E}[R]+\beta\mathbb{E}[E]$ [9, 15, 46], like the set of potential policies the set of potential cost functions is infinite. A subset of cost functions is referred to as *well-formed cost functions* and is defined as follows.

Definition 4.2. ***Well-Formed Cost Function:*** *A cost function, $\mathcal{C}(\cdot)$, is a well-formed cost function if it is non-decreasing in, dependent on, and only dependent on, the expected response time, i.e. $\mathbb{E}[R]$, and the expected energy cost, i.e. $\mathbb{E}[E]$. Moreover, a cost function, $\mathcal{C}(\cdot)$, is a linear well-formed cost function, if it is a well-formed cost function and linear in $\mathbb{E}[R]$ and $\mathbb{E}[E]$.*

With the previous notions of an energy-aware system, a policy, and a cost function in mind, one can view the problem illustrated by Figure 4.1 in finer detail in Figure 4.3. While associating formal parameters and metrics to the previous problem overview

gives a more concrete understanding of how to follow this method, some fundamental problems are still present. Specifically, it can be unclear how or why one would *choose* a policy (especially before any analysis has been done), and furthermore, how or why one would *choose* a well-formed cost function once that analysis has been done.

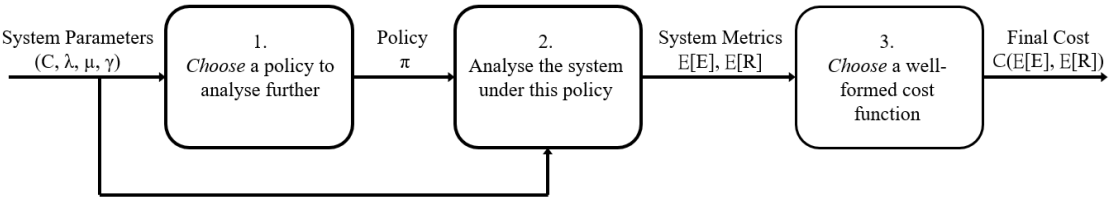


Figure 4.3: A broad overview of this thesis' problem and approach in finer detail

As an example, consider two policies π_1 and π_2. Arbitrarily, let π_1 be the policy that when possible, the number of servers in setup equals the number of jobs waiting in the queue, and has servers turn off the moment they idle. Also arbitrarily, let π_2 be the policy that when possible, the number of servers in setup equals the number of jobs in the system (not just waiting), and servers turn off if there are ever five or more servers idle. Even to the initiated neither of these policies seem unreasonable. In fact, it could be argued that it would be unreasonable to simply estimate (based on one's experience with these models) which policy one should employ between the two. However, the analysis of either of these policies is far from trivial. Moreover, computations needed to arrive at numerical values for $\mathbb{E}[R]$ and $\mathbb{E}[E]$ can be time consuming, especially for systems with larger values of C. Therefore, performing an analysis of both π_1 and π_2 may be infeasible. Taking a step back and realizing that the actual choice of policies is not just between π_1 and π_2, but in fact across an infinite

set, the choice can become daunting. As such, one is often forced to *guess* at which policy, or family of policies, will be fruitful to study.

Worse yet, this problem compounds when one then realizes they must also choose a cost function to optimize against. Returning to the two policies, one can note that π_2 turns and keeps servers on more readily than π_1 does. As such it may be reasonable to suggest that π_2 has a relatively low $\mathbb{E}[R]$ but a relatively high $\mathbb{E}[E]$. Conversely, π_1 has a relatively high $\mathbb{E}[R]$ but a relatively low $\mathbb{E}[E]$. Now consider the following two cost functions:

$$\mathcal{C}_1(\mathbb{E}[R], \mathbb{E}[E]) = \mathbb{E}[R] + 100\mathbb{E}[E] \quad \text{and} \quad \mathcal{C}_2(\mathbb{E}[R], \mathbb{E}[E]) = 100\mathbb{E}[R] + \mathbb{E}[E].$$

In other words, $\mathcal{C}_1$ values policies with a low expected energy cost, while $\mathcal{C}_2$ values policies with a low expected response time. As such under $\mathcal{C}_1$, π_1 may be deemed the better policy, while under $\mathcal{C}_2$, π_2 may be deemed the better policy. Therefore, it seems unreasonable to state one policy is strictly better than another. Again, this problem becomes ever more daunting when one realizes that there are not just two, but rather an infinite number of cost functions to choose from.

This problem complexity makes it seemingly impossible to make overarching claims, such as, for all energy-aware systems, policy π is optimal under all well-formed cost functions. To contradict this assertion, all one would need to do is identify a case in which π under-performs, e.g. short setup times, light load, etc., and suggest another policy specifically geared to handle that case, or inspect the metrics of π, and construct a cost function skewed to exploit π's (potentially niche) weakness. The

problem worsens as it is not even clear if one can make overarching exclusions, such as, for all energy-aware systems, policy π is suboptimal under all well-formed cost functions. It is entirely plausible that one could always manufacture an energy-aware system and/or cost function specifically tailored to magnify any unique, albeit slight, advantage which π may grant.

While the above arguments are valid, there exist approaches and measures one can take to confidently make broad, strong statements. In fact, this thesis will prove that in certain contexts for all energy-aware systems, many policies are optimal under all well-formed cost functions. This becomes more intuitive after one becomes more familiar with these systems and begins to realize that viewing a policy as a *tradeoff* between $\mathbb{E}[R]$ and $\mathbb{E}[E]$ is fundamentally incorrect, and insidiously misleading. That is, one can construct policies which simultaneously have low expected response times, and low expected energy cost. Essentially, in regards to efficacy and efficiency, policies exist which win on both fronts. Determining whether these policies exist, and moreover, which policies these are, takes some care.

This thesis addresses this problem using a three-pronged approach. Firstly, it addresses the infinite set of candidate policies to analyse further. This is done by examining the optimal policy. Specifically, by deriving structural properties the optimal policy is known to have, one can then eliminate all policies which do not exhibit this structure from the set of candidate policies, since such a policy would be known to always be suboptimal. Once several properties have been proven, the set of candidate policies will have been reduced to a significant degree. As such, a policy from this now

reduced set is guaranteed to share a number of behaviours with the optimal policy, and therefore has a greater chance of being a policy worth analysing further. This concept is illustrated in Figure 4.4.

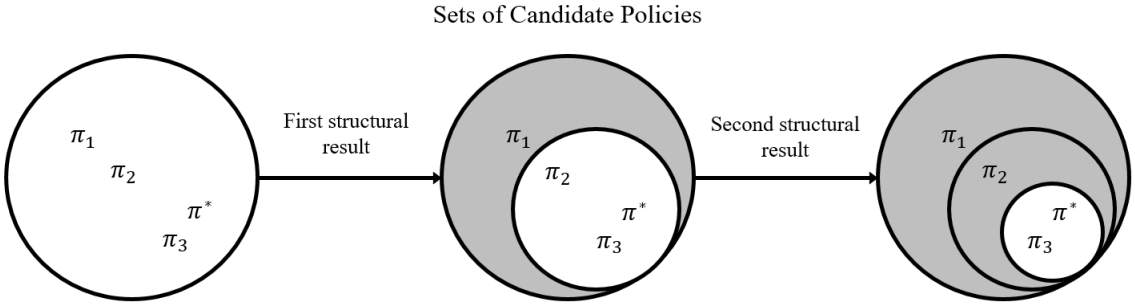


Figure 4.4: Reduction of the set of candidate policies via deriving optimal structure. Grey sections represent policies which are known to always be sub-optimal.

Secondly, once the set of candidate policies has been reduced, specific policies are selected to analyse further. Due to the underlying assumptions of the model, an exact analysis can be performed via several methods. Once analysed, one can then run numerical experiments to identify these aforementioned win-win scenarios where $\mathbb{E}[R]$ and $\mathbb{E}[E]$ are simultaneously minimized (or are reasonably close to being simultaneously minimized). From these identified cases, one can extrapolate these observations to provide general insights into how these systems should be provisioned and managed.

Lastly, this work examines what can be said when these systems become large. That is, the model is examined under a fixed-load, many-server asymptotic regime, where $C \to \infty$ while the load remains fixed, i.e. $0 < \rho < 1$. As will be seen in Chapter 7 when this is done a broad range of policies simultaneously minimize $\mathbb{E}[R]$ and $\mathbb{E}[E]$,

and therefore are optimal under all well-formed cost functions.

4.2 Analytic Tools

To achieve the goals described by the approach, a number of analytic tools are employed. Because the underlying distributions are all exponential, i.e. the interarrival, service, and setup times, Markovian methods can be applied. This work focuses on two. The first is a Markov decision processes (MDP) approach. Here, informally, the state space is a three-tuple from which one can precisely determine the number of jobs in the system, the number of servers currently on, and the number of servers in setup; the action space is the number of servers one wishes to turn on or off; decisions are made at every event, where an event is a job arriving or completing, or a server completing its setup process.

Using an MDP has advantages and disadvantages. In order to run well-known algorithms such as *policy iteration*, the state space must be finite. However, based on the definition of the model, specifically the presence of an infinite queue, the state space is infinite. Therefore, one must truncate the state space, which offers only an approximation of the optimal policy. If the truncation is done appropriately, said approximation may be reasonable or even equivalent to the optimal. What MDPs do allow one to achieve, without any compromise or truncation, is to examine the optimal policy in detail, and prove structural properties which it adheres to. As such, MDPs are leveraged when reducing the state space of candidate policies, as shown in Figure 4.4. The definition of an MDP of the model, as well as the optimal structural properties themselves are given in detail in Chapter 5.

The second Markovian tool used is continuous time Markov chains (CTMC). Unlike with MDPs, one can perform an exact analysis on a CTMC if it has an infinite state space. That is, one can theoretically arrive at closed form expressions for the cost metrics $\mathbb{E}[R]$ and $\mathbb{E}[E]$ without any compromise (truncation) to the definition of the model. Here the state space is an ordered pair, the first entry being the number of servers currently on, and the second being the number of jobs in the system; the state transitions are dependent on which policy is being employed. Details of these CTMCs, as well as their exact analysis, are presented in Chapter 6. An example of a CTMC corresponding to a smaller, two server, system is illustrated in Figure 4.5.

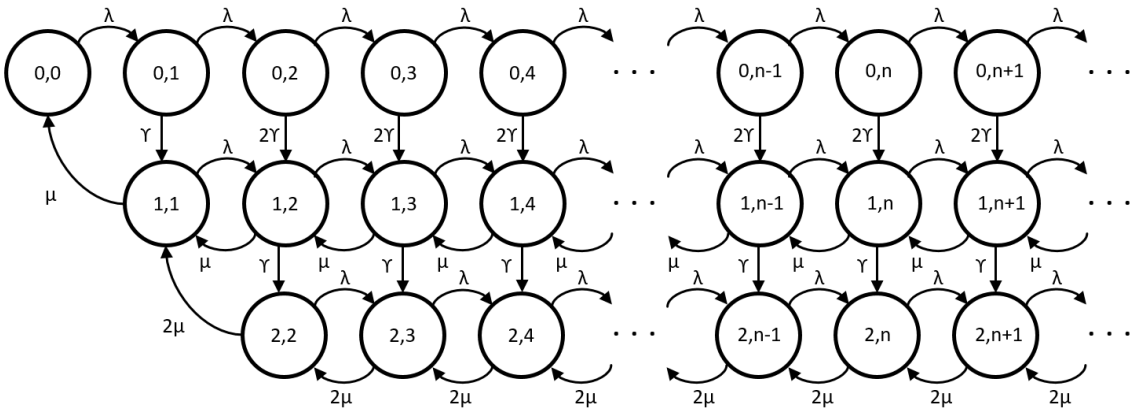


Figure 4.5: An example CTMC for a two server system where servers turn off when they idle, and the number in setup equals the number of waiting jobs when possible.

Table 4.2: Chapter 4 notation summary

Notation	Description
C	The number of servers
λ	The arrival rate; arrivals follow a Poisson process
μ	The service rate; service times are exponentially distributed
γ	The setup rate; setup times are exponentially distributed
$\mathbb{E}[R]$	The expected response time of the system
$\mathbb{E}[E]$	The expected energy cost of the system,
π	The employed policy, π^* is an optimal policy

4.3 Summary

The general approach to the problem is to reduce the set of candidate policies by deriving structural properties of the optimal policy via MDPs. From this reduced set, select policies to analyse further. Perform an exact analysis of these selected policies via CTMCs. From the exact analysis derive the metrics. Using the derived metrics, make insightful observations and conclusions across the problem domain. Formally back up these conclusions by analysing the asymptotic performance of the system, and show a large number of policies are optimal under all well-formed cost functions. A table of notation presented in this chapter is given in Table 4.2.

Chapter 5

Structural Properties

As will be seen, the optimal policy for energy-aware systems (if such a policy exists) can in general be quite complex and difficult to determine explicitly. However, there exist several structural properties which the optimal policy adheres to. These properties allow one to make decisions regarding when to turn servers on/off in a much more intelligent manner. Specifically, some decisions which may appear to be intuitively viable are shown to be suboptimal. This chapter is primarily reserved for presentation, discussion, and implications of the results, all proofs can be found in Appendix A.

Before any formal analysis can be performed on the optimal policy, a formal framework to determine the optimal policy (and structure therein) must first be defined. As such, Markov Decision Processes (MDPs) are employed to describe the control of an energy-aware system $S = (C, \lambda, \mu, \gamma)$. Without loss of generality it is assumed S has undergone uniformization, i.e. the rates (including those of dummy events) sums to one in all states. This work borrows the notation of [59] where A_s denotes the allowable set of actions in state s, and $p(s'|s, a)$ denotes the probability of being in

state s' at the next decision epoch, given that at the current decision epoch the system is in state s and performs action a. Here, an action a (the control) is the number of servers to turn on (or off if a is negative). Due to the nature of MDPs, here the state space must be a three-tuple rather than the two-tuple described in Chapter 4. That is, the state space is denoted by (i, j, m), where i, j, and m denote the number of servers on, jobs in the system, and servers in setup, respectively. Therefore, letting s, and s' be shorthand for (i, j, m) and (i', j', m') respectively, the first of two MDP formulations of an energy-aware system is given below,

$$p(s'|s,a) = \begin{cases} \lambda & \text{if} \quad j' = j+1,\ i' = i + \min(0,a), \\ & \qquad m' = m + \max(0,a) \\ \min(j, i + \min(0, m+a))\mu & \text{if} \quad j' = j-1,\ i' = i + \min(0,a), \\ & \qquad m' = m + \max(0,a) \\ (m + \max(0,a))\gamma & \text{if} \quad j' = j,\ i' = i + 1 + \min(0,a), \\ & \qquad m' = m - 1 + \max(0,a) \\ 1 - \lambda & \text{if} \quad j' = j,\ i' = i + \min(0,a), \\ - \min(j, i + \min(0,a))\mu & \qquad m' = m + \max(0,a) \\ -(m + \max(0,a))\gamma & \end{cases} \tag{5.3}$$

where,

$$A_{(i,j,m)} = \{-i, -i+1, ..., -1, 0, 1, ..., C-i-1, C-i\},$$

and the immediate cost (reward) function is given by $w(s, a)$, which for now is left unspecified. A key assumption of this formulation is that setups cannot be interrupted. One can infer this from the probabilities and action space. Formally, in the energy state *setup*, a server cannot be moved to the energy state *off*. For the majority of this thesis this assumption is relaxed (it is assumed setups can be interrupted) but due to this chapters' specific interest in the structure of the optimal policy, this assumption is imposed for a portion of the analysis. As such to make an explicit distinction, an energy-aware system where server setups cannot be interrupted is referred to as a *non-interruptible* energy-aware system, while a system where setups can be interrupted is referred to as an *interruptible* energy-aware system. For the remainder of this chapter, if a statement is made for energy-aware systems, it is implicitly stating it for both non-interruptible and interruptible energy-aware systems.

Fortunately, the MDP for an interruptible energy-aware system is significantly simpler. Due to the underlying exponential distribution of the setup times and their ability to be cancelled without penalty, the state space can be reduced from three parameters, to two. The MDP for an interruptible energy-aware system is given by:

$$p((i',j')|(i,j),a) = \begin{cases} \lambda & \text{if} \quad j' = j+1,\ i' = i + \min(0,a) \\ \min(j, i + \min(0,a))\mu & \text{if} \quad j' = j-1,\ i' = i + \min(0,a) \\ \max(0,a)\gamma & \text{if} \quad j' = j,\ i' = i+1 \\ 1 - \lambda \\ -\min(j, i + \min(0,a))\mu & \text{if} \quad j' = j,\ i' = i + \min(0,a) \\ -\max(0,a)\gamma \end{cases} \tag{5.4}$$

and

$$A_{(i,j)} = \{-i, -i+1, ..., -1, 0, 1, ..., C-i-1, C-i\},$$

and again the immediate cost (reward) function $w(s,a)$, is left unspecified. Note that at each decision epoch the decision variable a (if $a \geq 0$) represents the number of servers which will be in setup until the next decision epoch. This implies if one wished to cancel all current setup processes one would let $a = 0$. If $a < 0$ this implies no servers will be in setup (all current setups are interrupted) and $|a|$ of the on servers will be switched off.

These MDP formulations were a numerical starting point for all of the structural

properties presented in this chapter, the majority of formal proofs are performed independent from these formulations. Nevertheless these formulations still have inherent worth when reasoning about the structure and behaviour of the optimal policy.

To confidently grasp the more advanced structural properties of the optimal policy, it is imperative to first understand the rudiments of how the optimal policy behaves. In a Markovian setting like the one defined, if an optimal action is performed, it is immediately following a non-dummy event. This means that it is only optimal to turn off or on a server the moment a job arrives, a job leaves, or a server finishes its setup. The optimal policy also has two other simple structural properties that allow for its analysis to be somewhat simplified before more sophisticated properties are considered.

Theorem 5.1. *For all energy-aware systems, the optimal policy is a* pure policy. *That is, at every decision epoch, the optimal policy will never turn a non-zero number of servers off and at the same time put a non-zero number of servers into setup.*

This theorem is particularly convenient as an observant reader may have noticed this result embedded in the MDPs as an assumption (since there is only one action variable). This along with the next result allows one to begin to formally describe the optimal policy.

Theorem 5.2. *For all energy-aware systems, the decision to turn a specific server on or a specific server off follows a threshold policy based on the number of jobs in the system.*

This means that if in state (i, j, m) it is optimal to have m' servers in setup, then in state $(i, j + 1, m)$ it is optimal to have at least m' servers in setup. Likewise when

dealing with turning servers off, if it is optimal to turn off n servers in state (i, j, m), then it is also optimal to turn off at least n servers in state $(i, j-1, m)$.

Theorem 5.2 allows for a convenient description of the optimal policy, by simply providing a set of threshold values. For $0 \leq i < m \leq C$, the mth server will have a threshold value $k^+_{m,i}$ which indicates that if there are at least $k^+_{m,i}$ jobs in the system while there are i servers currently on, and if the mth server is currently off or in setup, then it will switch to or remain in setup. Conversely, if there are less than $k^+_{m,i}$ jobs in the system and the mth server is currently off, then it will remain off. Moreover, in an interruptible energy-aware system, if there are less than $k^+_{m,i}$ jobs in the system and the mth server is currently in setup, then it will be switched off. These values are referred to as the on-thresholds.

Furthermore, let $k^-_{i,m}$ denote the threshold decision variables for server turn-offs, where $0 < i \leq C$ and $0 \leq m < C - i$; the variable $k^-_{i,m}$ is the lower threshold number of jobs needed to turn a server off if there are exactly i servers on and m servers in setup. That is, in state (i, j, m) the ith server will be turned off if and only if $j \leq k^-_{i,m}$. These values are referred to as the off-thresholds. For an interruptible energy-aware system, the subscript m can be dropped as servers are free to move between setup and off and only servers which are already on need be considered when switching servers off. Therefore, for an interruptible energy-aware system, the off-thresholds are defined such that if in state (i, j) the ith server will be switched off if and only if $j \leq k^-_i$. From the definition of the threshold variables an initial ordering immediately follows.

Corollary 5.1. *For all $m, i \geq 0$ such that $m+i < C$, $k^+_{m,i} \leq k^+_{m+1,i}$ and $k^+_{m,i} \leq k^+_{m,i+1}$.*

Furthermore, for all $m \geq 0$ *and* $i > 0$ *such that* $m + i < C$, $k^-_{i,m} \leq k^-_{i+1,m}$. *For an interruptible energy-aware system for all* i *such that* $0 \leq i < C$, $k^-_i \leq k^-_{i+1}$.

For an energy-aware system $S = (C, \lambda, \mu, \gamma)$, to find the optimal policy one must determine $C(C+1)/2+C$ threshold values if S is interruptible and $C(C+1)$ threshold values if it is not. Delving deeper into these threshold values leads to a result which offers a significant reduction to the state space which should be considered.

Theorem 5.3. *For all non-interruptible energy-aware systems, if while in the state* $(i-1, j, m)$ *it is optimal to begin turning a server on, then in state* (i, j, m) *it is sub-optimal to turn a server off. Furthermore, for all interruptible energy-aware systems if while in the state* $(i-1, j)$ *it is optimal to begin turning a server on, then in state* (i, j) *it is suboptimal to turn a server off.*

Corollary 5.2. *For all non-interruptible energy-aware systems, if while in the state* (i, j, m) *it is optimal to turn a server off, then for all* $j' \leq j$ *it is suboptimal to turn a server on in state* $(i-1, j', m)$. *Furthermore, for all interruptible energy-aware systems if while in the state* (i, j) *it is optimal to turn a server off, then for all* $j' \leq j$ *it is suboptimal to turn a server on in state* $(i-1, j')$.

Corollary 5.3. *For all* $0 < m \leq C$ *and* $0 \leq i < m$, $k^-_{i,m} < k^+_{m,i}$, *and* $k^-_i < k^+_{m,i}$.

For a system with instantaneous setups, i.e. $1/\gamma = 0$, this theorem becomes trivial, since it would make no sense to turn a server on and then immediately turn it off. However, consider the more interesting case of a system with long setup times ($1/\gamma$ is large) in some state (i, j, m) where in state $(i+1, j, m)$ it is known that turning a server off is optimal. It may be reasonable to think that although the system was in a state where it is optimal to turn a server off given it was currently turned on,

it might still be optimal to *begin* turning it on in anticipation of many jobs arriving before the server completes its setup. However, this is not the case. Importantly, this gives a relation between the orderings of the *off* and *on* thresholds, where before there was none. From here a comprehensive partial order for all threshold values can be derived.

Theorem 5.4. *For all* $0 \leq i < (C-1)$ *and* $0 \leq j < (C-i-1)$, $k^+_{m+1,i} \leq k^+_{m,i+1}$, *and* $0 < i < C$ *and* $0 \leq j \leq (C-i)$, $k^-_{i,m+1} \leq k^-_{i+1,m}$ *and* $k^-_{i,m} \leq k^-_{i,m+1}$.

Corollary 5.4. *A partial order can be defined on the threshold values, of which the lattice is shown in Figure 5.1, where* $x \to y$ *implies* $x \leq y$.

One should note that in the optimal policy it is possible to have some number of servers which always remain on. If this is the case, a system employing an optimal policy in state $(C, 0, 0)$ would immediately turn off $C - c^*$ servers, where c^* is the optimal number of servers which always remain on. Figure 5.1 still captures this behaviour, by letting $k^+_{m,i} \leq 0$ for all $m + i < c^*$, alongside $k^-_{i,m} < 0$ for all $i + m \leq c^*$. From Theorem 5.3 and the fact that there cannot be less than zero jobs in the system, it is known these servers will always remain on (once they have been turned on).

Theorem 5.3 offers insight into on-thresholds relative to the off-thresholds. Therefore, it would be appealing to gain direct insight into the off-thresholds themselves. A popular simplification when creating tractable models for these systems is to have servers turn off as soon as they idle. In general, when describing the optimal policy, there is no compelling reason why this should be the case. In fact, feasible values for off-thresholds seem to be as free as the on-thresholds. Clearly, there exists a configuration of parameters and cost function where it is optimal to have a server idle until

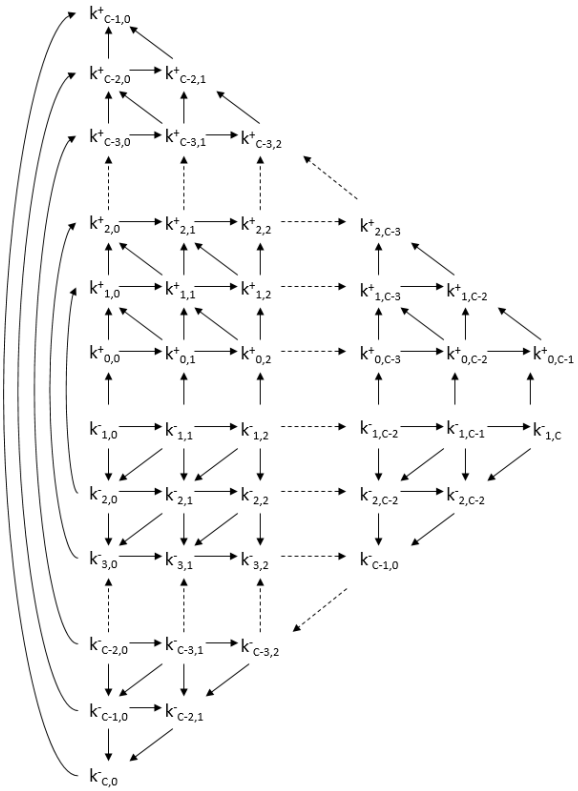


Figure 5.1: Threshold lattice: some orderings (arrows) from Theorem 5.3 are left off for readability

some lower threshold is reached (a server which always remains on is a trivial example of this). But conversely, is there a configuration of parameters and cost function where it is optimal to turn a server off while there are jobs to be processed?

At first glance there seems to be two arguments which suggest that there should be. Firstly, turning a server off will decrease the energy consumed by the system in the short run (although it will increase holding costs). Secondly, and the more subtle of the two arguments, keeping a server on (referred to as s_1) will cause the system to remove jobs at faster rate. Thus, other servers (referred to as s_2 and s_3) could start

to turn off sooner due to jobs departing quickly. Once s_2, s_3 and potentially other servers are off, more jobs could arrive and cause those servers to now start a setup which could have been avoided if s_1 was initially turned off instead of allowing for jobs to be processed at a fast rate. In essence, the system could thrash. Therefore, if s_1 was instead turned off the system would process jobs at a slower rate, which in turn causes less setups, and correspondingly less potential energy costs. These arguments seem to indicate that the cost tradeoffs in the optimal policy may be inordinately complex. However, while these arguments may seem intuitive, they are in fact incorrect, as demonstrated by the next theorem.

Theorem 5.5. *For all energy-aware systems, for all linear well-formed cost functions, if the number of jobs in the system is greater than or equal to the number of servers currently turned on, it is always suboptimal to turn a server off, which is already on.*

Corollary 5.5. *For all linear well-formed cost functions, for all* $1 \leq i \leq C$ *and* $0 \leq j < C - i$, $k^-_{i,m} < i$ *and* $k^-_i < i$.

With regards to the fallacious intuition presented before Theorem 5.5, it would never make sense to turn a server off while there is a job to process for the sole reason of saving the energy cost to process the job. The job will have to be processed eventually, otherwise the system is not stable, or unnecessary holding costs are incurred. Therefore, it is preferable to incur the energy cost now, since processing it earlier will also decrease the holding cost. It is much harder to show the second argument (keeping a server on in the short run causes more setups in the long run) to be false, and this is where the real value of Theorem 5.5 is seen. One of the popular turn off policies used in the literature is to turn a server off the moment it idles, such as the "staggered setup" model in [14]. Theorem 5.5 also allows such models to be used with

greater confidence as it is now shown that such a policy is not suboptimal with respect to the turn off criterion. On the other hand, if one were to construct a policy such that a server will turn off when there are $k > 0$ jobs waiting in the queue, it would immediately be known to be suboptimal. While it is true that the above only holds for a subset (albeit a popular subset) of well-formed cost functions, it is conjectured that this holds for all well-formed cost functions.

Theorem 5.5 gives concrete guidance as to when to turn servers off. However, it remains to determine an explicit structure pertaining to turning servers on. As such, this work introduces the notion of a *bulk setup* scheme. Here, when the system decides to turn a server on, instead of just putting one server in setup, it instead puts all available servers into setup. Assuming the system in question is an interruptible energy-aware system, it follows that once one of the servers completes its setup, the system can simultaneously cancel all remaining setups if it chooses to. One may see this as incurring unnecessary costs, but it offers advantages as well. In particular, where before there were $C(C+1)/2$ on-thresholds, the bulk setup scheme decreases this to C. If there are i servers currently on, where $0 \leq i < C$, then there is a single threshold k_i^+ such that if there are greater than or equal to k_i^+ jobs in the system then the remaining $(C-i)$ servers are in setup. Conversely, if there are less than k_i^+ jobs in the system then the remaining $(C-i)$ are off. A fortunate result of this scheme is that under an appropriate cost function, there is no disadvantage.

Theorem 5.6. *For all interruptible energy-aware systems, for all well-formed cost functions linear in* $\mathbb{E}[R]$ *and* $\mathbb{E}[E]$*, the optimal policy turns servers on following a bulk setup scheme.*

While perhaps surprising at first, this flavour of result arises in other research [8] and follows from the assumption of exponentially distributed setup times. If one were to turn m servers on, although the energy costs would be incurred at m times the rate of the single server, due to the nature of the exponential distribution, it would be for an expected amount of time that is reduced by a factor m. Compare this to having the setup time follow a degenerate distribution (constant setup times). Under this assumption such a policy would be a disaster, especially if C were large.

While the sensitivity to the distribution can be worrisome, there is another potential problem with such a policy. In practice, managers are reluctant to turn machines on/off due to potential wear-and-tear, risk of failure, etc. As such, some authors incorporate a switching cost into their cost function associated with such an action [9, 40, 44]. Clearly, for such a cost function a bulk setup policy would not be optimal in general. While the bulk setup issue can be addressed by incorporating switching into the cost function, one could instead constrain the model directly to not allow such behaviour. This is seen in models where the turn on policy follows some predefined structure, such as the staggered setup described in [20]. But if one chooses to exclude switching from the cost function and allows freedom with regard to turn on decisions, they should be aware of this problem when searching for an optimal policy.

With these notions in mind it seems that when setups are interruptible, the assumption of exponentially distributed setup times alongside a cost function independent of switching costs gives rise to a model with optimal behaviour that may be problematic to actually implement. Therefore, one should be cautious when using results

derived from such a model. These concerns are considered in Chapter 6, but it will be seen that using a bulk setup policy as a base, one can make powerful conclusions for determining near optimal policies which do not use a bulk setup scheme.

5.1 Three Server Example

Here a small example is presented to demonstrate the worth of the structural properties with regards to reduction of search space. Consider an energy-aware system $S = (3, \lambda, \mu, \gamma)$ where $\lambda < 3\mu$. Firstly, from Theorems 5.1 and 5.2 it is known that an optimal policy has twelve decision variables. These decision variables are the on-threshold $k^+_{m,i}$ where $m, i \geq 0$ and $m + i < 3$, and the off-thresholds $k^-_{i,m}$ where $i > 0$, $m \geq 0$ and $i + m \leq 3$.

When examining the off-thresholds, if in complete ignorance of previously presented results in this chapter, then the set of feasible values is $\{-1, 0, 1, 2, ..., K-1, K\}$ for an arbitrarily large constant K. However, applying Theorem 5.5 (assuming a linear well-formed cost function), the feasible state space shrinks significantly. The feasible sets of values for the off thresholds $k^-_{1,m}$, $k^-_{2,m}$, and $k^-_{3,m}$ are $\{-1, 0\}$, $\{-1, 0, 1\}$ and $\{-1, 0, 1, 2\}$, respectively. This result alone makes the problem much easier to tackle as one has greater confidence in heuristics for picking such values, as well as the ability to iterate through a much smaller set of values if determining the optimal policy numerically.

Changing focus to the on-thresholds, some simplifications can also be made. Similar to the off-thresholds, with no structural results the set of potential thresholds has K

elements, i.e. $\{0, 1, 2, ..., K-1, K\}$ for some arbitrarily large constant K. While the presented theorems cannot be applied directly, simplifications can be made assuming the off-thresholds are known. Specifically, using the off-thresholds the set of feasible values can be truncated by applying Theorem 5.3. That is, for each threshold $k^+_{m,i}$ the feasible set of values now becomes $\{k^-_{i+1,m}+1, k^-_{i+1,m}+2, k^-_{i+1,m}+3, ..., K-1, K\}$. While the truncation only eliminates a relatively small number of values from the set, they are arguably the most important values to eliminate. The greater the value of the threshold, the lesser the impact increasing (or decreasing) that threshold by one would have on the performance of the system. For example in general, if the optimal value of say $k^+_{2,0}$ were 3 and a value of 2 was chosen, the difference in performance would be greater than say if the optimal value was 20 and 19 was chosen. Therefore, even a small truncation could have a large impact when choosing these thresholds.

Although reduction in the search space for specific threshold values is useful, the main advantage of these structural properties is applying them to policy heuristics to ensure said heuristics are not suboptimal. This allows for a confident and feasible analysis of energy-aware systems, even when C is large.

Chapter 6

Exact Analysis

Once a policy for an energy-aware system has been determined, it remains to analyse said policy. Due to Definition 4.2, to properly evaluate a policy one must be able to arrive at the values for $\mathbb{E}[R]$ and $\mathbb{E}[E]$. In general, this is an intractable task for large values of C (there being a combinatorial explosion of decision variables). When using the results presented in Chapter 5 however, one can restrict the set of policies such that the analysis of $\mathbb{E}[R]$ and $\mathbb{E}[E]$ has sufficient structure to be tractable. Specifically, from Theorem 5.2, limiting the scope of policies to threshold policies allows for all policies, for all energy-aware systems, to be modelled as a two dimensional CTMC with the following two useful properties:

1. The state space of the CTMC is (i, j), where i denotes the number of servers on, and j denotes the number of jobs in the system. Therefore, the CTMC is infinite in one dimension (the number of jobs), but finite in the other (the number of servers on). Moreover, a state can only transition to states adjacent to it. That is, if in state (i, j), the next state the system will be in is one of $(i + 1, j)$, $(i - 1, j)$, $(i, j + 1)$, or $(i, j - 1)$. As an aside, this is referred to

as a quasi-birth-death process (QBD), and is well understood in the stochastic modelling community.

2. There exists a column j such that the CTMC begins repeating for all columns greater than j. That is, there exists a j^* such that for all i and $j \geq j^*$ if the transition rates to states $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, $(i, j-1)$ from state (i, j) equal r_1, r_2, r_3, and r_4 respectively, then the transition rates to states $(i+1, j+1)$, $(i-1, j+1)$, $(i, j+2)$, (i, j) from state $(i, j+1)$ also equal r_1, r_2, r_3, and r_4, respectively. As an example, in the CTMC illustrated in Figure 6.1 the repeating portion of the chain begins at column 6.

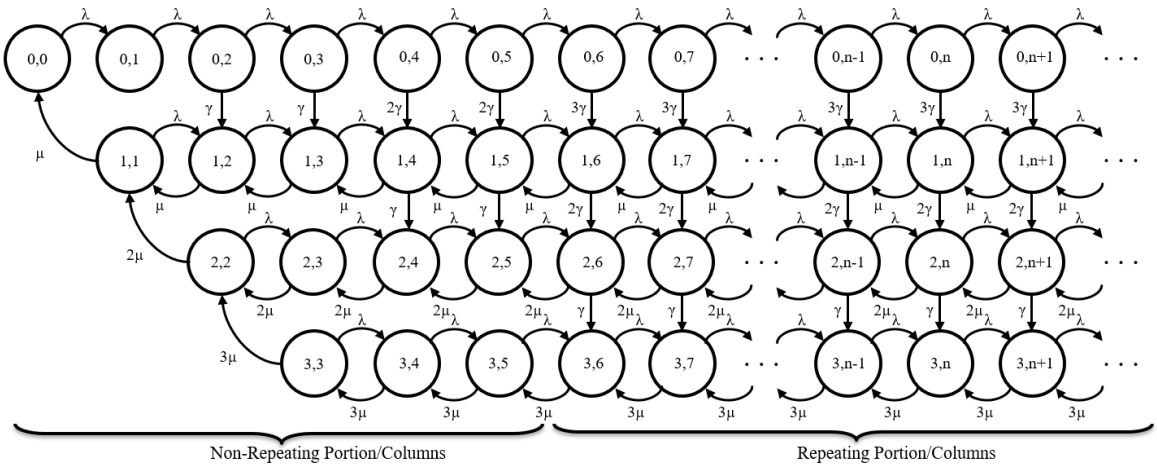


Figure 6.1: The underlying CTMC for an energy-aware system $(3, \lambda, \mu, \gamma)$, implementing a threshold policy.

With this structure, a variety of methods exist to analyse the underlying CTMC. For example, two of the more popular methods are z-transforms, and the matrix geometric method. This thesis uses the recursive renewal reward (RRR) method, which exploits the structure of the CTMC to build a finite recursion, and then leverages the renewal

reward theorem (given in Chapter 2) to arrive at the cost metrics, i.e. $\mathbb{E}[R]$ and $\mathbb{E}[E]$. To understand this method, some notation and concepts must be introduced. In general, the RRR method keeps track of how much of a certain *reward* is accumulated before the CTMC arrives to the column directly left of the current column. That is, it tracks the reward accumulated when the CTMC has j jobs present until the moment it has $j-1$.

For the purposes of this work, the derived rewards (hereafter referred to as costs) are the expected amount of time, the expected holding costs (at rate one per job), and the expected excess energy costs incurred before transitioning one column left. For state (i,j) these values are denoted by $T_{i,j}$, $H_{i,j}$ and $E_{i,j}$, respectively. As a reminder to the definition of excess energy, because jobs must be processed eventually, the choice of (stable) policy has no impact on how much energy is spent processing jobs (busy servers). What the policy does have impact on, is the amount of energy spent idling and setting up servers. For ease of exposition, for the remainder of chapter, excess energy will simply be referred to as energy.

Due to the assumption of stability, i.e. $\lambda < C\mu$, all of these values of $T_{i,j}$, $H_{i,j}$ and $E_{i,j}$ are finite. As a visual aid, in Figure 6.1, $T_{1,3}$ would denote the expected amount of time for the system to reach one of the states $(0,2)$, $(1,2)$, or $(2,2)$, given that it started in state $(1,3)$. The value $H_{0,5}$ would denote the expected amount of holding cost incurred during the time the system transitions from state $(0,5)$ to one of the states $(0,4)$, $(1,4)$, $(2,4)$, or $(3,4)$. Furthermore, to build a recursive relationship between these values, one must know the probability of being in a particular state once a

transition one column left has been made. Therefore, $P_{i'}(i,j)$ denotes the probability of being in row i' after moving one column left of state (i,j). In Figure 6.1, $P_2(0,4)$ would denote the probability of being in state $(2,3)$ the moment the system reaches one of the states $(0,3)$, $(1,3)$, $(2,3)$, or $(3,3)$, given it started in state $(0,4)$. The recursions for these costs and probabilities are "tied off" once the CTMC reaches the repeating portion.

6.1 Repeated Values

One of the fortunate properties of these systems is that regardless of which threshold policy is being implemented, the repeating portions of the CTMC are equivalent. Therefore, one can analyse the repeating portion of these CTMCs with complete generality. Hence, that is where the analysis of these CTMCs begins.

6.1.1 Repeated Row Probabilities

Due to the nature of the CTMC in the repeating portion, one can write down quadratic equations which describe the column transition probabilities. Firstly, consider the value $P_C(C,j)$ where j is large enough to be in the repeating portion of the chain. This is trivially known, since the system must first progress to the left before it can progress down the rows (the threshold to turn the Cth server off is outside the repeating portion). Therefore $P_C(C,j) = 1$. For ease of expression, since the transition probabilities are independent of which column the system is in once it is in the repeating portion, the j parameter is suppressed. Therefore, the previous equality can be rewritten as $P_C(C) = 1$. The more interesting cases arise when the system is

not in row C. Progressing through the rows one inspects the terms $P_C(C-1)$ and $P_{C-1}(C-1)$. Again it is known that when transitioning one column left the system will either remain in its current row, or move up some number of rows. Therefore, $P_C(C-1) + P_{C-1}(C-1) = 1$. The explicit expressions are derived as follows:

$$P_{C-1}(C-1) = \frac{(C-1)\mu}{\lambda + (C-1)\mu + \gamma} + \frac{\lambda}{\lambda + (C-1)\mu + \gamma} P_{C-1}^2(C-1)$$

$$\Rightarrow \quad 0 = \frac{\lambda}{\lambda + (C-1)\mu + \gamma} P_{C-1}^2(C-1) - P_{C-1}(C-1) + \frac{(C-1)\mu}{\lambda + (C-1)\mu + \gamma}$$

$$\begin{aligned} P_C(C-1) = &\frac{\gamma}{\lambda + (C-1)\mu + \gamma} \\ &+ \frac{\lambda}{\lambda + (C-1)\mu + \gamma}(P_C(C-1) + P_{C-1}(C-1)P_C(C-1)) \end{aligned}$$

$$\Rightarrow \quad P_C(C-1)\left(1 - \frac{\lambda}{\lambda + (C-1)\mu + \gamma}(1 + P_{C-1}(C-1))\right) = \frac{\gamma}{\lambda + (C-1)\mu + \gamma}$$

$$\Rightarrow \quad P_C(C-1) = \frac{\gamma}{(C-1)\mu + \gamma - \lambda P_{C-1}(C-1)}$$

Here it is seen that solving for $P_{C-1}(C-1)$ yields a quadratic, while solving for $P_C(C-1)$ does not. For the general expression $P_{i'}(i)$, where $0 \leq i, i' \leq C$, it can be said that if $i' = i$ then the resulting expression is quadratic, but otherwise is not. However, it is noted that if $i' < i$ then $P_{i'}(i) = 0$ due to the structure of the CTMC (to move from row i to $i-1$, the system must first move at least one column left).

This is explicitly given below.

$$P_i(i) = \frac{i\mu}{\lambda + i\mu + (C-i)\gamma} + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma} P_i^2(i)$$

$$\Rightarrow \quad 0 = \frac{\lambda}{\lambda + i\mu + (C-i)\gamma} P_i^2(i) - P_i(i) + \frac{i\mu}{\lambda + i\mu + (C-i)\gamma} \tag{6.5}$$

The above quadratic yields real roots. Furthermore, the stability condition of $\lambda < C\mu$ implies a unique steady state distribution, which guarantees exactly one root lies between 0 and 1. The analysis proceeds under the assumption $i' > i$ (recall if $i' < i$, then it is trivially known $P_{i'}(i) = 0$).

$$P_{i'}(i) = \frac{(C-i)\gamma}{\lambda + i\mu + (C-i)\gamma} P_{i'}(i+1) + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma} \sum_{m=i}^{i'} P_{i'}(m) P_m(i)$$

$$\Rightarrow \quad P_{i'}(i) = \frac{(C-i)\gamma P_{i'}(i+1) + \lambda \sum_{m=i+1}^{i'-1} P_{i'}(m) P_m(i)}{i\mu + (C-i)\gamma + \lambda(1 - P_i(i) - P_{i'}(i'))} \tag{6.6}$$

6.1.2 Repeated Transition Time Values

In order to apply the RRR method to derive a desired metric, an expected cycle time must always be known. To derive an expected cycle time, the expected column transition time must be known. As in Section 6.1.1, because it is assumed that the system is in a column which is in the repeating portion of the chain, the column denotation of $T_{i,j}$ is suppressed. Therefore, T_i denotes the expected time to move one column left, given the system is in the repeating portion of the chain and in row i. Fortunately, unlike the repeated probabilities, solving these values does not result in

a quadratic:

$$T_i = \frac{1}{i\mu + (C-i)\gamma + \lambda} + \frac{(C-i)\gamma}{i\mu + (C-i)\gamma + \lambda} T_{i+1} + \frac{\lambda}{i\mu + (C-i)\gamma + \lambda}\left(T_i + \sum_{i'=i}^{C} T_{i'} P_{i'}(i)\right)$$

$$\Rightarrow \quad T_i\left(\frac{i\mu + (C-i)\gamma - \lambda P_i(i)}{i\mu + (C-i)\gamma + \lambda}\right) = \frac{1 + (C-i)\gamma}{i\mu + (C-i)\gamma + \lambda} T_{i+1} + \frac{\lambda}{i\mu + (C-i)\gamma + \lambda} \sum_{i'=i+1}^{C} T_{i'} P_{i'}(i)$$

$$\Rightarrow \quad T_i = \frac{1 + (C-i)\gamma T_{i+1} + \lambda \sum_{i'=i+1}^{C} T_{i'} P_{i'}(i)}{i\mu + (C-i)\gamma - \lambda P_i(i)}. \tag{6.7}$$

While the above equation is valid for $i = C$, it is also noted that T_C is equivalent to the busy period of an M/M/1 queue with service rate $C\mu$. That is,

$$T_C = \frac{1}{C\mu - \lambda}.$$

6.1.3 Repeated Cost Values

One of the metrics this analysis aims to derive is the expected number of jobs in the system, i.e. $\mathbb{E}[N]$. Recall from Little's law, $\mathbb{E}[R]$ immediately follows from knowledge of $\mathbb{E}[N]$. The approach is to first calculate the expected holding cost incurred over a single cycle, and then divide that by the expected cycle time. From the renewal reward theorem, this ratio is known to equal $\mathbb{E}[N]$. Just as when solving for the cycle time, one must solve the column transition values with respect to the holding cost

incurred. To solve for the transition values, the repeating portion must be solved. However, unlike repeated probability and time values, the holding costs of states in the same row are not equal in the repeating portion, i.e. for $j \geq j^*$, $H_{i,j} \neq H_{i,j+1}$. Fortunately, for $j \geq j^*$ an expression for $H_{i,j}$ can be derived which is not dependent on values from other columns, by noting that $H_{i,j+1} = H_{i,j} + T_i$.

$$H_{i,j} = \frac{j}{\lambda + i\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + i\mu + (C-i)\gamma} H_{i+1,j} + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma}\left(H_{i,j+1} + \sum_{m=i}^{C} H_{m,j} P_m(i)\right) \tag{6.8}$$

$$\Rightarrow \quad H_{i,j}\left(\frac{i\mu + (C-i)\gamma - \lambda P_i(i)}{\lambda + i\mu + (C-i)\gamma}\right) = \frac{j}{\lambda + i\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + i\mu + (C-i)\gamma} H_{i+1,j} + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma}\left(T_i + \sum_{m=i+1}^{C} H_{m,j} P_m(i)\right)$$

$$\Rightarrow \quad H_{i,j} = \frac{j + (C-i)\gamma H_{i+1,j} + \lambda(T_i + \sum_{m=i+1}^{C} H_{m,j} P_m(i))}{i\mu + (C-i)\gamma - \lambda P_i(i)} \tag{6.9}$$

The top row also deserves a special mention here as it is always repeating for $j \geq C$ (if Theorem 5.5 is being enforced) and can be used as a base case when solving the non-repeating portion. For $j \geq C$,

$$H_{C,j} = \frac{j}{C\mu - \lambda} + \frac{\lambda}{C\mu - \lambda} H_{C,j+1}$$

$$\Rightarrow \quad H_{C,j} \frac{C\mu}{C\mu - \lambda} = \frac{j}{C\mu - \lambda} + \frac{\lambda}{C\mu - \lambda} T_C$$

$$\Rightarrow \quad H_{C,j} = \frac{j + \lambda T_C}{C\mu}$$

Now all that remains to solve for in the repeating portion are the energy consumption values.

$$E_{i,j} = \frac{(C-i)E_{\text{Setup}}}{\lambda + i\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + i\mu + (C-i)\gamma} E_{i+1,j} + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma}\left(E_{i,j+1} + \sum_{m=i}^{C} E_{m,j} P_m(i)\right) \tag{6.10}$$

$$\Rightarrow \quad E_{i,j}\left(\frac{i\mu + (C-i)\gamma - \lambda P_i(i)}{\lambda + i\mu + (C-i)\gamma}\right) = \frac{(C-i)E_{\text{Setup}}}{\lambda + i\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + i\mu + (C-i)\gamma} E_{i+1,j} + \frac{\lambda}{\lambda + i\mu + (C-i)\gamma} \sum_{m=i+1}^{C} E_{m,j} P_m(i)$$

$$\Rightarrow \quad E_{i,j} = \frac{(C-i)E_{\text{Setup}} + (C-i)\gamma E_{i+1,j} + \lambda \sum_{m=i+1}^{C} E_{m,j} P_m(i)}{i\mu + (C-i)\gamma - \lambda P_i(i)} \tag{6.11}$$

Thus, the repeated transition values for all threshold policies are now solved, and can be used to terminate the recursion for specific policy implementations.

6.2 Bulk Setup

The first specific policy is the *bulk setup* policy. For any policy to be fully defined one must determine what criteria must be true for each server to turn on, and what criteria must be true for a server to turn off. A formal definition of this policy is given below. Recall that static servers are those which always remain on, and dynamic

servers are those which may switch on and off.

Definition 6.1. ***Bulk Setup Policy:*** *A policy is a bulk setup policy if it has two decision variables, C_S and k, where C_S denotes the number of servers which always remain on, i.e the number of static servers, and k is a threshold variable such that dynamic servers behave in the following manner:*

- *For all m and i where $C_S < m \leq C$, and $C_S \leq i < m$, $k^+_{m,i} = C_S + (i - C_S + 1)k$. In other words, if there are ever $C_S + (i - C_S + 1)k$ or more jobs in the system, where i is the number of servers currently on, then all remaining servers will be in setup, and if the number of jobs in the system is less than $C_S + (i - C_S + 1)k$, then all servers which are not already on, will be switched off (have their setups cancelled).*
- *For all m where $C_S < m \leq C$, $k^-_m = m - 1$. That is, servers turn off the moment they idle.*

Figure 6.2 gives an underlying CTMC for an energy-aware system implementing a bulk setup policy.

6.2.1 Boundary Probabilities

Before transition probabilities for the non-repeating portion can be derived, it is important to note that certain boundary probabilities are known. For example, it is known that in Figure 6.2 $P_3(4, 4) = 1$, since once the system progresses to column 3 (there are three jobs in the system), all idle servers will immediately turn off.

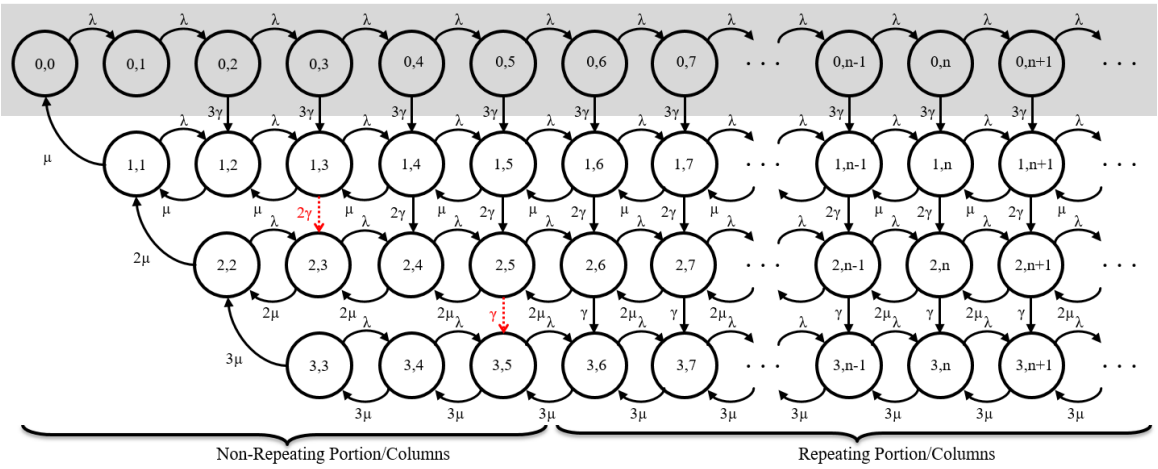


Figure 6.2: The underlying CTMC for an energy-aware system $(3, \lambda, \mu, \gamma)$, implementing a bulk setup policy with $C_S = 0$, and $k = 2$. If the parameters were to be changed to $C_S = 1$, and $k = 2$, row 0 (shaded over in grey) would be merged with row 1, and the red dotted line (red) transitions would be added.

Therefore,

$$P_i(i+1, i+1) = 1.$$

More formally, one can express all boundary probabilities as follows.

$$(\forall i > C_S : \ P_{i-1}(i, i) = P_{i-1}(i-1, i) = 1)$$
$$(\forall j \leq C_S : \ P_{C_S}(C_S, j) = 1)$$

It is assumed that all future derivations of the non-repeating transition probabilities in this section are not boundary probabilities.

6.2.2 Row Probabilities

All metrics hinge on the probabilities corresponding to which row the CTMC will be in once it has transitioned one column left of its current state. Therefore, the analysis starts there. Firstly, the row C_S probabilities are solved. The C_Sth row can be broken into two parts, where the column j is such that all remaining servers are off, or all remaining servers are turning on, i.e. $j - C_S < k$ and $j - C_S \geq k$. Beginning with $j - C_S < k$ and $j > 0$, the following is derived.

$$P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu} + \frac{\lambda}{\lambda + \min(C_S, j)\mu} P_{C_S}(C_S, j) P_{C_S}(C_S, j+1)$$

$$\Rightarrow \quad P_{C_S}(C_S, j)\left(1 - \frac{\lambda}{\lambda + \min(C_S, j)\mu} P_{C_S}(C_S, j+1)\right) = \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu}$$

$$\Rightarrow \quad P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu}{\min(C_S, j)\mu + \lambda(1 - P_{C_S}(C_S, j+1))}$$

The analysis proceeds with solving $P_{C_S+1}(C_S, j)$.

$$P_{C_S+1}(C_S, j) = \frac{\lambda}{\lambda + \min(C_S, j)\mu}\Big((P_{C_S+1}(C_S, j) P_{C_S}(C_S, j+1) + P_{C_S+1}(C_S+1, j)(P_{C_S+1}(C_S, j+1))\Big)$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j)\left(1 - \frac{\lambda}{\lambda + \min(C_S, j)\mu} P_{C_S}(C_S, j+1)\right) = \frac{\lambda}{\lambda + \min(C_S, j)\mu} P_{C_S+1}(C_S+1, j) P_{C_S+1}(C_S, j+1)$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j) = \frac{\lambda P_{C_S+1}(C_S+1, j) P_{C_S+1}(C_S, j+1)}{\min(C_S, j)\mu + \lambda(1 - P_{C_S}(C_S, j+1))}$$

From here, an argument can be made that for any column transition, to get to row i' when moving one column left of state (C_S, j), the probability is the product of the probabilities of the column transition to get to row m from state $(C_S, j+1)$ and the probability of the column transition of getting to row i' from state (m, j). This is assuming $i' \neq C_S$, in which case an extra term would need to be added, equal to the probability of a departure being the next event witnessed by the system.

$$P_{i'}(C_S, j) = \frac{\lambda}{\lambda + \min(C_S, j)\mu} \sum_{m=C_S}^{i'} P_{i'}(m, j) P_m(C_S, j+1)$$

$$\Rightarrow \quad P_{i'}(C_S, j) = \frac{\lambda \sum_{m=C_S+1}^{i'} P_{i'}(m, j) P_m(C_S, j+1)}{\min(C_S, j)\mu + \lambda(1 - P_{C_S}(C_S, j+1))}$$

The portion of the C_S row where $j - C_S \geq k$ is a different story as the next event may be a setup completion. Proceeding as before (solving for $P_{C_S}(C_S, j)$ and $P_{C_S+1}(C_S, j)$) gives rise to a slightly different pattern.

$$\begin{aligned} P_{C_S}(C_S, j) = {} & \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} \\ & + \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S}(C_S, j) P_{C_S}(C_S, j+1) \end{aligned}$$

$$\Rightarrow \quad P_{C_S}(C_S, j)\left(1 - \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S}(C_S, j+1)\right)$$

$$= \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma}$$

$$\Rightarrow \quad P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu}{\min(C_S, j)\mu + (C - C_S)\gamma + \lambda(1 - P_{C_S}(C_S, j+1))}$$

Proceeding with the next row yields the following.

$$\begin{aligned} P_{C_S+1}(C_S, j) = {} & \frac{(C - C_S)\gamma}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S+1}(C_S + 1, j) \\ & + \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} \Big(P_{C_S+1}(C_S, j) P_{C_S}(C_S, j+1) \\ & \quad + P_{C_S+1}(C_S + 1, j) P_{C_S+1}(C_S + 1, j+1) \Big) \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad & P_{C_S+1}(C_S, j) \Big(1 - \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S}(C_S, j+1) \Big) \\ & = \frac{(C - C_S)\gamma}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S+1}(C_S + 1, j) \\ & \quad + \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{C_S+1}(C_S + 1, j) P_{C_S+1}(C_S + 1, j+1) \end{aligned}$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j) = \frac{(C - C_S)\gamma P_{C_S+1}(C_S + 1, j) + \lambda P_{C_S+1}(C_S + 1, j) P_{C_S+1}(C_S, j+1)}{\min(C_S, j)\mu + (C - C_S)\gamma + \lambda(1 - P_{C_S}(C_S, j+1))}$$

Moving to a general i'th row from this portion of the chain proceeds as before, but now with a γ term present in the numerator.

$$\begin{aligned} P_{i'}(C_S, j) = {} & \frac{(C - C_S)\gamma}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} P_{i'}(C_S + 1, j) \\ & + \frac{\lambda}{\lambda + \min(C_S, j)\mu + (C - C_S)\gamma} \sum_{m=C_S}^{i'} P_{i'}(m, j) P_m(C_S, j+1) \end{aligned}$$

$$\Rightarrow \quad P_{i'}(C_S, j) = \frac{(C - C_S)\gamma P_{i'}(C_S + 1, j) + \lambda \sum_{m=C_S+1}^{i'} P_{i'}(m, j) P_m(C_S, j + 1)}{\min(C_S, j)\mu + (C - C_S)\gamma + \lambda(1 - P_{C_S}(C_S, j + 1))}$$

This completes all the column transition probabilities from row C_S. That is, the probabilities of being in one of the rows C_S through C once the system moves one column left, given it began in row C_S. However, it remains to calculate the transition probabilities given that the system started in an arbitrary row. In other words, it remains to derive $P_{i'}(i, j)$ for all valid values of i, j, and i'. Similar to the above derivations, the analysis starts with the simpler case of $i' = i$. However, no matter which valid row is under consideration, the two distinct parts of the non-repeating portion must be considered separately. That is, the portions of the ith row where $j < (i - C_S + 1)k + C_S$ and where $j \geq (i - C_S + 1)k + C_S$. The analysis starts with the simpler case of $j < (i - C_S + 1)k + C_S$.

$$P_i(i, j) = \frac{\min(i, j)\mu}{\lambda + \min(i, j)\mu} + \frac{\lambda}{\lambda + \min(i, j)\mu} P_i(i, j) P_i(i, j + 1)$$

$$\Rightarrow \quad P_i(i, j) = \frac{\min(i, j)\mu}{\min(i, j)\mu + \lambda(1 - P_i(i, j + 1))}$$

The analysis proceeds under the assumption $j < (i - C_S + 1)k + C_S$, but relaxes the restriction on i', i.e. now $i' > i$.

$$P_{i'}(i, j) = \frac{\lambda}{\lambda + \min(i, j)\mu} \sum_{m=i}^{i'} P_{i'}(m, j) P_m(i, j + 1)$$

$$\Rightarrow \quad P_{i'}(i,j) = \frac{\lambda \sum_{m=i+1}^{i'} P_{i'}(m,j) P_m(i,j+1)}{\min(i,j)\mu + \lambda(1 - P_i(i,j+1))}$$

This fully solves for all of the column transition probabilities for $j < (i - C_S + 1)k + C_S$. The case where transitions due to setups are present is now considered, i.e. $j \geq (i - C_S + 1)k + C_S$. Firstly, assuming $i' = i$,

$$\begin{aligned} P_i(i,j) = & \frac{\min(i,j)\mu}{\lambda + \min(i,j)\mu + (C-i)\gamma} \\ & + \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma} P_i(i,j) P_{C_S+i}(i,j+1) \end{aligned}$$

$$\Rightarrow \quad P_i(i,j) = \frac{\min(i,j)\mu}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}.$$

Secondly, assuming $i' > i$,

$$\begin{aligned} P_{i'}(i,j) = & \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma} P_{i'}(i+1,j) \\ & + \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma} \sum_{m=i}^{i'} P_{i'}(m,j) P_m(i,j+1) \end{aligned}$$

$$\Rightarrow \quad P_{i'}(i,j) = \frac{(C-i)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{i'} P_{i'}(m,j) P_m(i,j+1)}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}.$$

With the above solved, the probability derivations are now complete.

6.2.3 Transition Costs

To apply the renewal reward theorem one must look at the cost incurred over a single cycle of any system state. The most basic (and also necessary) cost to solve for is the cycle time. One can view this as the system incurring cost at a rate of one per unit of time. With the transition probabilities solved for, deriving the expected amount of time to move one column left of the current state is relatively simple. As was done with the non-repeating portion of each row i for the transition probabilities, the time values for each row must be solved for the two distinct parts. That is when the column $j < (i-C_S+1)k+C_S$, and when it is past its setup threshold, or $j \geq (i-C_S+1)k+C_S$. Firstly, the former is assumed, i.e. $j < (i - C_S + 1)k + C_S$.

$$T_{i,j} = \frac{1}{\lambda + \min(i,j)\mu} + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(T_{i,j+1} + \sum_{m=i}^{C} T_{m,j}P_m(i,j+1)\right)$$

$$\begin{aligned}\Rightarrow \quad T_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu}\right) &= \frac{1}{\lambda + \min(i,j)\mu} \\ &\quad + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j}P_m(i,j+1)\right)\end{aligned}$$

$$\Rightarrow \quad T_{i,j} = \frac{1 + \lambda(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j}P_m(i,j+1))}{\min(i,j)\mu + \lambda(1 - P_i(i,j+1))}$$

The analysis proceeds under the assumption $j \geq (i - C_S + 1)k + C_S$.

$$\begin{aligned}T_{i,j} = \frac{1}{\lambda + \min(i,j)\mu + (C-i)\gamma} &+ \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma}T_{i+1,j} \\ &+ \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}\left(T_{i,j+1} + \sum_{m=i}^{C} T_{m,j}P_m(i,j+1)\right)\end{aligned}$$

$$\Rightarrow \quad T_{i,j}\left(1 - \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma} P_i(i,j+1)\right)$$
$$= \frac{1}{\lambda + \min(i,j)\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma} T_{i+1,j}$$
$$+ \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}\left(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1)\right)$$

$$\Rightarrow \quad T_{i,j} = \frac{1 + (C-i)\gamma T_{i+1,j} + \lambda(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1))}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}$$

With the expected times solved for, one can proceed with the derivation of expressions pertaining to the cost metrics. As mentioned previously, these are the expected response time of the system, and the expected rate of energy consumption.

Solving for the expected response time is equivalent to solving for the expected number of jobs in the system, due to Little's law. To solve for the expected number of jobs in the system, the expected accumulated holding cost incurred while in a non-repeating state (i,j) before transitioning to column $j-1$, is derived. It is worth noting that this derivation is similar to that of $T_{i,j}$, with the difference being cost is incurred at a rate equal to the number in the system, or j, rather than at a rate of one. As before, the non-repeating portion of each row must be analysed separately. Again, the analysis begins by assuming $j < (i - C_S + 1)k + C_S$.

$$H_{i,j} = \frac{j}{\lambda + \min(i,j)\mu} + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(H_{i,j+1} + \sum_{m=i}^{C} H_{m,j} P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu}\right) = \frac{j}{\lambda + \min(i,j)\mu} + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j} = \frac{j + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1))}{\min(i,j)\mu + \lambda(1 - P_i(i,j+1))}$$

Now assume that there are enough jobs present to begin turning the remaining servers on, i.e. $j \geq (i - C_S + 1)k + C_S$.

$$H_{i,j} = \frac{j}{\lambda + \min(i,j)\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma}H_{i+1,j} + \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}\left(H_{i,j+1} + \sum_{m=i}^{C} H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j}\left(1 - \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}P_i(i,j+1)\right) = \frac{j}{\lambda + \min(i,j)\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma}H_{i+1,j} + \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}\left(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j} = \frac{j + (C-i)\gamma H_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1))}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}$$

Lastly, the analysis derives $E_{i,j}$, which denotes the expected amount of energy accumulated by starting in state (i,j) before transitioning to column $j-1$. As before, it is firstly assumed that $j < (i - C_S + 1)k + C_S$. However, these distinct portions

of the rows tell one a bit more when solving for the expected energy. Firstly, if $j < (i - C_S + 1)k + C_S$ then it is known that no servers are currently in setup, and therefore no immediate consequential energy costs are accumulated. Furthermore, it is in the definition of the policy that a server will turn off the moment it idles. This implies that no energy costs are immediately incurred, unless the system is in row C_S, where the policy allows for servers to idle.

$$E_{i,j} = \frac{\max(i-j,0)E_{\text{Idle}}}{\lambda + \min(i,j)\mu} + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(E_{i,j+1} + \sum_{m=i}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu}\right) = \frac{\max(i-j,0)E_{\text{Idle}}}{\lambda + \min(i,j)\mu} + \frac{\lambda}{\lambda + \min(i,j)\mu}\left(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j} = \frac{\max(i-j,0)E_{\text{Idle}} + \lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1))}{\min(i,j)\mu + \lambda(1 - P_i(i,j+1))}$$

Proceeding under the assumption that $j \geq (i - C_S + 1)k + C_S$,

$$E_{i,j} = \frac{(C-i)E_{\text{Setup}}}{\lambda + \min(i,j)\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma}E_{i+1,j} + \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}\left(H_{i,j+1} + \sum_{m=i}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j}\left(1 - \frac{\lambda}{\lambda + \min(i,j)\mu + (C-i)\gamma}P_i(i,j+1)\right) = \frac{(C-i)E_{\text{Setup}}}{\lambda + \min(i,j)\mu + (C-i)\gamma} + \frac{(C-i)\gamma}{\lambda + \min(i,j)\mu + (C-i)\gamma}E_{i+1,j}$$

$$+\frac{\lambda}{\lambda+\min(i,j)\mu+(C-i)\gamma}\left(E_{i,j+1}+\sum_{m=i+1}^{C}E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow\quad E_{i,j}=\frac{(C-i)E_{\text{Setup}}+(C-i)\gamma E_{i+1,j}+\lambda(E_{i,j+1}+\sum_{m=i+1}^{C}E_{m,j}P_m(i,j+1))}{\min(i,j)\mu+(C-i)\gamma+\lambda(1-P_i(i,j+1))}.$$

6.2.4 Summary

Here all final closed form equations needed to implement the method are listed. Firstly, the boundary values for the transition probabilities,

$$(\forall i>C_S:\ P_{i-1}(i,i)=1)\quad\text{and}\quad(\forall j\leq C_S:\ P_{C_S}(C_S,j)=1). \tag{6.12}$$

All required expressions pertaining to the non-repeating portion of the chain where no servers are in setup, i.e. when $j<(i-C_S+1)k+C_S$, $C_S\leq i\leq C$, and $i\leq i'$ are as follows:

$$P_{i'}(i,j)=\frac{\lambda\sum_{m=i+1}^{i'}P_{i'}(m,j)P_m(i,j+1)}{\min(i,j)\mu+\lambda(1-P_i(i,j+1))}, \tag{6.13}$$

$$T_{i,j}=\frac{1+\lambda(T_{i,j+1}+\sum_{m=i+1}^{C}T_{m,j}P_m(i,j+1))}{\min(i,j)\mu+\lambda(1-P_i(i,j+1))}, \tag{6.14}$$

$$H_{i,j}=\frac{j+\lambda(H_{i,j+1}+\sum_{m=i+1}^{C}H_{m,j}P_m(i,j+1))}{\min(i,j)\mu+\lambda(1-P_i(i,j+1))}, \tag{6.15}$$

$$E_{i,j}=\frac{\max(i-j,0)E_{\text{Idle}}+\lambda(E_{i,j+1}+\sum_{m=i+1}^{C}E_{m,j}P_m(i,j+1))}{\min(i,j)\mu+\lambda(1-P_i(i,j+1))}. \tag{6.16}$$

Finally, all required expressions are compiled for the non-repeating portion of the chain where servers are in setup, i.e. when $j \geq (i - C_S + 1)k + C_S$, $C_S \leq i \leq C$, and $i \leq i'$. They are as follows:

$$P_{i'}(i,j) = \frac{(C-i)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{i'} P_{i'}(m,j) P_m(i,j+1)}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.17}$$

$$T_{i,j} = \frac{1 + (C-i)\gamma T_{i+1,j} + \lambda(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1))}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.18}$$

$$H_{i,j} = \frac{j + (C-i)\gamma H_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j} P_m(i,j+1))}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.19}$$

$$E_{i,j} = \frac{(C-i)E_{\text{Setup}} + (C-i)\gamma E_{i+1,j} + \lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j} P_m(i,j+1))}{\min(i,j)\mu + (C-i)\gamma + \lambda(1 - P_i(i,j+1))}. \tag{6.20}$$

6.3 Dual Threshold

While the setup portion of the policy in Section 6.2 is known to be optimal for linear well-formed cost functions, it is admittedly unappealing to implement in practice, not true if switching costs are included, and furthermore, dependent on the assumption imposed on the setup times being exponentially distributed. Therefore, an alternate policy is analysed.

Definition 6.2. ***Dual Threshold Policy:*** *A policy is a dual threshold policy if it has two decision variables, C_S and k, where C_S denotes the number of servers which always remain on, i.e the number of static servers, and k is a threshold variable such*

that dynamic servers behave in the following manner:

- *For all m and i where $C_S < m \leq C$, and $C_S \leq i < m$, $k^+_{m,i} = (m - C_S)k$. This implies that if there are i servers on, and j jobs in the system, the number of servers in setup is given by $f(i,j) = \{\lfloor j/k \rfloor - (i - C_S)\}^+$.*

- *For all m and i, where $C_S < m \leq C$ and $C_S \leq i < m$, $k^-_m = \min(k^+_{m,i} - 1, m - 1)$. That is, a server turns off the moment the number of jobs drops below said server's setup threshold* ***and*** *said server is also idle.*

One can note that the system is less aggressive with regards to its setup behaviour than a bulk setup policy. That is, servers are turned on gradually as needed instead of all at once. As mentioned previously in Section 6.1, all the derived repeated values still apply to this policy. However, the column in which this policy starts repeating is $(C - C_S)k$.

6.3.1 Boundary Probabilities

As for the bulk setup policy, one must keep track of the boundary states where a server will turn off once the system progresses to one column left of that state. In the bulk setup policy this was easily done due to the structure of a server turning off if and only if it is idle. However, as described in the previous section, in this policy it is possible for a server to remain idle for a time before it turns off (because its off threshold is reached). Therefore, these boundaries require some special attention. Specifically, the boundary transition probability for row i is,

$$P_{i-1}(i,j) = 1, \quad \text{where} \quad j = \min(i, k(i - C_S)).$$

As for the bulk setup policy, it is assumed that all of the following derived transition probabilities are not boundary values.

6.3.2 Row Probabilities

To build up the recursions, the analysis proceeds in the same manner as was done in Section 6.2. As such, beginning with the row transition probabilities.

$$P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{\lambda P_{C_S}(C_S, j) P_{C_S}(C_S, j+1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{f(C_S, j)\gamma}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} P_{C_S}(C_S + 1, j)$$

$$\Rightarrow \quad P_{C_S}(C_S, j)\left(1 - \frac{\lambda P_{C_S}(C_S, j+1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}\right) = \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{f(C_S, j)\gamma}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} P_{C_S}(C_S + 1, j)$$

$$\Rightarrow \quad P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu + f(C_S, j)\gamma P_{C_S}(C_S + 1, j)}{\min(C_S, j)\mu + f(C_S, j)\gamma + \lambda(1 - P_{C_S}(C_S, j+1))}$$

The analysis continues by deriving $P_{C_S+1}(C_S, j)$.

$$P_{C_S+1}(C_S, j) = \frac{f(C_S, j)\gamma P_{C_S+1}(C_S + 1, j)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{\lambda(P_{C_S+1}(C_S, j) P_{C_S}(C_S, j+1) + P_{C_S+1}(C_S + 1, j) P_{C_S+1}(C_S, j+1))}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j)\left(1 - \frac{\lambda P_{C_S}(C_S, j+1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}\right)$$

$$= \frac{f(C_S,j)\gamma P_{C_S+1}(C_S+1,j)}{\lambda+\min(C_S,j)\mu+f(C_S,j)\gamma} + \frac{\lambda P_{C_S+1}(C_S+1,j)P_{C_S+1}(C_S,j+1)}{\lambda+\min(C_S,j)\mu+f(C_S,j)\gamma}$$

$$\Rightarrow \quad P_{C_S+1}(C_S,j) = \frac{f(C_S,j)\gamma P_{C_S+1}(C_S+1,j)+\lambda P_{C_S}(C_S,j+1)}{\min(C_S,j)\mu+f(C_S,j)\gamma+\lambda(1-P_{C_S}(C_S,j+1))}$$

With the transition probabilities of the first two rows solved explicitly, the analysis proceeds to the general cases. Firstly, when $i' = i$,

$$P_i(i,j) = \frac{\min(i,j)\mu}{\lambda+\min(i,j)\mu+f(i,j)\gamma} + \frac{f(i,j)\gamma P_{i'}(i+1,j)}{\lambda+\min(i,j)\mu+f(i,j)\gamma} + \frac{\lambda P_{i'}(i,j)P_i(i,j+1)}{\lambda+\min(i,j)\mu+f(C_S,j)\gamma},$$

$$\Rightarrow \quad P_i(i,j)\left(1-\frac{\lambda P_i(i,j+1)}{\lambda+\min(C_S,j)\mu+f(C_S,j)\gamma}\right) = \frac{\min(i,j)\mu+f(i,j)\gamma P_{i'}(i+1,j)}{\lambda+\min(i,j)\mu+f(i,j)\gamma},$$

$$\Rightarrow \quad P_i(i,j) = \frac{\min(i,j)\mu+f(i,j)\gamma P_{i'}(i+1,j)}{\min(i,j)\mu+f(i,j)\gamma+\lambda(1-P_i(i,j+1))}.$$

Continuing with the general case, where $i' \neq i$,

$$P_{i'}(i,j) = \frac{f(i,j)\gamma P_{i'}(i+1,j)}{\lambda+\min(i,j)\mu+f(i,j)\gamma} + \frac{\lambda\sum_{m=i}^{C} P_{i'}(m,j)P_m(i,j+1)}{\lambda+\min(i,j)\mu+f(C_S,j)\gamma},$$

$$\Rightarrow \quad P_{i'}(i,j)\left(1-\frac{\lambda P_i(i,j+1)}{\lambda+\min(i,j)\mu+f(C_S,j)\gamma}\right) = \frac{f(i,j)\gamma P_{i'}(i+1,j)+\lambda\sum_{m=i+1}^{C} P_{i'}(m,j)P_m(i,j+1)}{\lambda+\min(i,j)\mu+f(i,j)\gamma},$$

$$\Rightarrow \quad P_{i'}(i,j) = \frac{f(i,j)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{C} P_{i'}(m,j) P_m(i,j+1)}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}.$$

6.3.3 Transition Costs

With the probabilities solved, the analysis proceeds with the transition costs for the time, holding, and energy consumption values.

$$\begin{aligned} T_{i,j} = {} & \frac{1}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma} T_{i+1,j} \\ & + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma} \left(T_{i,j+1} + \sum_{m=i}^{C} T_{m,j} P_m(i,j+1) \right) \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad T_{i,j} \left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma} \right) = {} & \frac{1}{\lambda + \min(i,j)\mu + f(i,j)\gamma} \\ & + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma} T_{i+1,j} \\ & + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma} \left(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1) \right) \end{aligned}$$

$$\Rightarrow \quad T_{i,j} = \frac{1 + f(i,j)\gamma T_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}$$

$$\begin{aligned} H_{i,j} = {} & \frac{i}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma} H_{i+1,j} \\ & + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma} \left(H_{i,j+1} + \sum_{m=i}^{C} H_{m,j} P_m(i,j+1) \right) \end{aligned}$$

$$\Rightarrow \quad H_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\right) = \frac{i + f(i,j)\gamma H_{i+1,j}}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\left(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j} = \frac{i + f(i,j)\gamma H_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j}P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}$$

$$E_{i,j} = \frac{\max(0, i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}}}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma}E_{i+1,j} + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\left(E_{i,j+1} + \sum_{m=i}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\right) = \frac{\max(0, i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}} + f(i,j)\gamma E_{i+1,j}}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\left(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j} = \frac{\max(0, i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}} + f(i,j)\gamma E_{i+1,j}}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))} + \frac{\lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}$$

6.3.4 Summary

In summary, the equations required to implement the RRR method for this policy are as follows. Firstly, set the boundary column transitions (when a server turns off).

$$\begin{aligned}(\forall i > C_S :\ & (\forall j \text{ such that } j = \min(i, (i - C_S)k) :\ P_{i-1}(i, j) = 1)) \\ \text{and}\quad & (\forall j \leq k :\ P_{C_S}(C_S, j) = 1).\end{aligned} \tag{6.21}$$

A list of the column transition values is given as follows, where the indices are given as $C_S \leq i \leq C$, $i \leq i'$, (i, j) is not a boundary state given in (6.21), and $j < \max(k(C - C_S), C_S)$, i.e. j is to the left of the repeating column:

$$P_i(i, j) = \frac{\min(i, j)\mu + f(i, j)\gamma P_{i'}(i + 1, j)}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))}, \tag{6.22}$$

$$P_{i'}(i, j) = \frac{f(i, j)\gamma P_{i'}(i + 1, j) + \lambda \sum_{m=i+1}^{C} P_{i'}(m, j) P_m(i, j + 1)}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))}, \tag{6.23}$$

$$T_{i,j} = \frac{1 + f(i, j)\gamma T_{i+1,j} + \lambda(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i, j + 1))}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))}, \tag{6.24}$$

$$H_{i,j} = \frac{i + f(i, j)\gamma H_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j} P_m(i, j + 1))}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))}, \tag{6.25}$$

$$\begin{aligned}E_{i,j} = & \frac{\max(0, i - j)E_{\text{Idle}} + f(i, j)E_{\text{Setup}} + f(i, j)\gamma E_{i+1,j}}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))} \\ & + \frac{\lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j} P_m(i, j + 1))}{\min(i, j)\mu + f(i, j)\gamma + \lambda(1 - P_i(i, j + 1))},\end{aligned} \tag{6.26}$$

where, as a reminder, $f(i,j) = \{\lfloor j/k \rfloor - i\}^+$.

6.4 Staggered Threshold

Here a third policy is introduced which incorporates aspects from the policies defined in Sections 6.2 and 6.3. This is done with the goal of combining favourable aspects of both previously described policies, while eliminating some potentially problematic ones. That is, the policy described here will not have the unappealing implementation issues of Bulk Setup, but will have more direct control over how many servers are kept operational.

Definition 6.3. ***Staggered Threshold Policy:*** *A policy is a staggered threshold policy if it has two decision variables, C_S and k, where C_S denotes the number of servers which always remain on, i.e the number of static servers, and k is a threshold variable such that dynamic servers behave in the following manner:*

- *For all m and i where $C_S < m \leq C$, and $C_S \leq i < m$, $k^+_{m,i} = C_S + (m - C_S)k$. This implies that if there are i servers on, and j jobs in the system, the number of servers in setup is given by $f(i,j) = \{\lfloor \{j - C_S\}^+/k \rfloor - (i - C_S)\}^+$.*
- *For all m where $C_S < m \leq C$, $k^-_m = m-1$. That is, servers turn off the moment they idle.*

This can again intuitively be thought of as each of the dynamic servers being responsible for k jobs when all of the static servers are busy. The difference from the bulk setup policy is this policy gradually begins to turn more servers on instead of all at once. For example, if all dynamic servers are off, and there are $C_S + k$ jobs in the

system exactly one server will be moved to setup, if there are $C_S + 2k$ jobs in the system, then two servers will be in setup, and so on.

For the sake of completeness the analysis proceeds following the pattern shown in Sections 6.2 and 6.3. In fact, due to the abstract definition of $f(\cdot)$, the analysis is identical to that in Section 6.3. Therefore, if the reader is comfortable with these derivations, they could skip ahead to Section 6.4.4 without loss of clarity.

6.4.1 Boundary Probabilities

As usual, a definition for all of the boundary transition states which are known to equal one is given. Since an idle server will instantly turn off, it is known that $P_i(i+1, i+1) = 1$. Furthermore, although covered in future derivations in this section, it is also trivially known that $P_i(i, i+1) = 1$. Again this follows from the fact that an idle server must always be switched off immediately, assuming it is not static.

6.4.2 Row Probabilities

The recursion is built from the base case, beginning with the probability of being in row C_S after moving one column left of state (C_S, j).

$$P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{\lambda P_{C_S}(C_S, j) P_{C_S}(C_S, j+1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{f(C_S, j)\gamma}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} P_{C_S}(C_S + 1, j)$$

$$\Rightarrow \quad P_{C_S}(C_S, j)\left(1 - \frac{\lambda P_{C_S}(C_S, j+1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}\right)$$

$$= \frac{\min(C_S, j)\mu}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{f(C_S, j)\gamma}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} P_{C_S}(C_S + 1, j)$$

$$\Rightarrow \quad P_{C_S}(C_S, j) = \frac{\min(C_S, j)\mu + f(C_S, j)\gamma P_{C_S}(C_S + 1, j)}{\min(C_S, j)\mu + f(C_S, j)\gamma + \lambda(1 - P_{C_S}(C_S, j + 1))}$$

Continuing by deriving $P_{C_S+1}(C_S, j)$,

$$P_{C_S+1}(C_S, j) = \frac{f(C_S, j)\gamma P_{C_S+1}(C_S + 1, j)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{\lambda(P_{C_S+1}(C_S, j)P_{C_S}(C_S, j + 1) + P_{C_S+1}(C_S + 1, j)P_{C_S+1}(C_S, j + 1))}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j)\left(1 - \frac{\lambda P_{C_S}(C_S, j + 1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}\right) = \frac{f(C_S, j)\gamma P_{C_S+1}(C_S + 1, j)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma} + \frac{\lambda P_{C_S+1}(C_S + 1, j)P_{C_S+1}(C_S, j + 1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}$$

$$\Rightarrow \quad P_{C_S+1}(C_S, j) = \frac{f(C_S, j)\gamma P_{C_S+1}(C_S + 1, j) + \lambda P_{C_S}(C_S, j + 1)}{\min(C_S, j)\mu + f(C_S, j)\gamma + \lambda(1 - P_{C_S}(C_S, j + 1))}.$$

With the transition probabilities to the first two rows solved explicitly, the analysis proceeds to the general cases. Firstly, when $i' = i$,

$$P_i(i, j) = \frac{\min(i, j)\mu}{\lambda + \min(i, j)\mu + f(i, j)\gamma} + \frac{f(i, j)\gamma P_{i'}(i + 1, j)}{\lambda + \min(i, j)\mu + f(i, j)\gamma} + \frac{\lambda P_{i'}(i, j)P_i(i, j + 1)}{\lambda + \min(i, j)\mu + f(C_S, j)\gamma}$$

$$\Rightarrow \quad P_i(i, j)\left(1 - \frac{\lambda P_i(i, j + 1)}{\lambda + \min(C_S, j)\mu + f(C_S, j)\gamma}\right) = \frac{\min(i, j)\mu + f(i, j)\gamma P_{i'}(i + 1, j)}{\lambda + \min(i, j)\mu + f(i, j)\gamma}$$

$$\Rightarrow \quad P_i(i,j) = \frac{\min(i,j)\mu + f(i,j)\gamma P_{i'}(i+1,j)}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}.$$

Now, with $i' \neq i$,

$$P_{i'}(i,j) = \frac{f(i,j)\gamma P_{i'}(i+1,j)}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{\lambda \sum_{m=i}^{C} P_{i'}(m,j) P_m(i,j+1)}{\lambda + \min(i,j)\mu + f(C_S,j)\gamma}$$

$$\Rightarrow \quad P_{i'}(i,j)\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(C_S,j)\gamma}\right) = \frac{f(i,j)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{C} P_{i'}(m,j) P_m(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma}$$

$$\Rightarrow \quad P_{i'}(i,j) = \frac{f(i,j)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{C} P_{i'}(m,j) P_m(i,j+1)}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}.$$

6.4.3 Transition Costs

Proceeding with the transition costs for the time, holding, and energy consumption values yields the following.

$$T_{i,j} = \frac{1}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma} T_{i+1,j} + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\left(T_{i,j+1} + \sum_{m=i}^{C} T_{m,j} P_m(i,j+1)\right)$$

$$\Rightarrow \quad T_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\right) = \frac{1}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{f(i,j)\gamma}{\lambda + \min(i,j)\mu + f(i,j)\gamma} T_{i+1,j}$$

$$+\frac{\lambda}{\lambda+\min(i,j)\mu+f(i,j)\gamma}\left(T_{i,j+1}+\sum_{m=i+1}^{C}T_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad T_{i,j}=\frac{1+f(i,j)\gamma T_{i+1,j}+\lambda(T_{i,j+1}+\sum_{m=i+1}^{C}T_{m,j}P_m(i,j+1))}{\min(i,j)\mu+f(i,j)\gamma+\lambda(1-P_i(i,j+1))}$$

$$H_{i,j}=\frac{i}{\lambda+\min(i,j)\mu+f(i,j)\gamma}+\frac{f(i,j)\gamma}{\lambda+\min(i,j)\mu+f(i,j)\gamma}H_{i+1,j}$$
$$+\frac{\lambda}{\lambda+\min(i,j)\mu+f(i,j)\gamma}\left(H_{i,j+1}+\sum_{m=i}^{C}H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j}\left(1-\frac{\lambda P_i(i,j+1)}{\lambda+\min(i,j)\mu+f(i,j)\gamma}\right)=\frac{i+f(i,j)\gamma H_{i+1,j}}{\lambda+\min(i,j)\mu+f(i,j)\gamma}$$
$$+\frac{\lambda}{\lambda+\min(i,j)\mu+f(i,j)\gamma}\left(H_{i,j+1}+\sum_{m=i+1}^{C}H_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad H_{i,j}=\frac{i+f(i,j)\gamma H_{i+1,j}+\lambda(H_{i,j+1}+\sum_{m=i+1}^{C}H_{m,j}P_m(i,j+1))}{\min(i,j)\mu+f(i,j)\gamma+\lambda(1-P_i(i,j+1))}$$

Since the only portion of the chain where idle servers are present is the C_Sth row, this can be accounted for in a single expression as follows.

$$E_{i,j}=\frac{\max(0,i-j)E_{\text{Idle}}+f(i,j)E_{\text{Setup}}}{\lambda+\min(i,j)\mu+f(i,j)\gamma}+\frac{f(i,j)\gamma}{\lambda+\min(i,j)\mu+f(i,j)\gamma}E_{i+1,j}$$
$$+\frac{\lambda}{\lambda+\min(i,j)\mu+f(i,j)\gamma}\left(E_{i,j+1}+\sum_{m=i}^{C}E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j}\left(1 - \frac{\lambda P_i(i,j+1)}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\right) = \frac{\max(0,i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}} + f(i,j)\gamma E_{i+1,j}}{\lambda + \min(i,j)\mu + f(i,j)\gamma} + \frac{\lambda}{\lambda + \min(i,j)\mu + f(i,j)\gamma}\left(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1)\right)$$

$$\Rightarrow \quad E_{i,j} = \frac{\max(0,i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}} + f(i,j)\gamma E_{i+1,j}}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))} + \frac{\lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j}P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}$$

6.4.4 Summary

In summary, the equations required to implement the RRR method for this policy are as follows. Firstly, set the boundary column transitions (when a server turns off).

$$P_i(i+1,i+1) = 1, \quad P_i(i,i+1) = 1 \tag{6.27}$$

A list of the column transition values are given as follows, where the indices are given as $C_S \leq i \leq C$, $i \leq i'$, (i,j) is not a boundary state given in (6.27), and $j < \max(k(C - C_S) + C_S, C_S)$, i.e. j is left of the repeating column:

$$P_i(i,j) = \frac{\min(i,j)\mu + f(i,j)\gamma P_{i'}(i+1,j)}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.28}$$

$$P_{i'}(i,j) = \frac{f(i,j)\gamma P_{i'}(i+1,j) + \lambda \sum_{m=i+1}^{C} P_{i'}(m,j)P_m(i,j+1)}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.29}$$

$$T_{i,j} = \frac{1 + f(i,j)\gamma T_{i+1,j} + \lambda(T_{i,j+1} + \sum_{m=i+1}^{C} T_{m,j} P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.30}$$

$$H_{i,j} = \frac{i + f(i,j)\gamma H_{i+1,j} + \lambda(H_{i,j+1} + \sum_{m=i+1}^{C} H_{m,j} P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}, \tag{6.31}$$

$$\begin{aligned} E_{i,j} = {} & \frac{\max(0, i-j)E_{\text{Idle}} + f(i,j)E_{\text{Setup}} + f(i,j)\gamma E_{i+1,j}}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))} \\ & + \frac{\lambda(E_{i,j+1} + \sum_{m=i+1}^{C} E_{m,j} P_m(i,j+1))}{\min(i,j)\mu + f(i,j)\gamma + \lambda(1 - P_i(i,j+1))}, \end{aligned} \tag{6.32}$$

where, as a reminder, $f(i,j) = \{\lfloor\{j - C_S\}^+/k\rfloor - (i - C_S)\}^+$.

6.5 Derivation of $\mathbb{E}[N]$ and $\mathbb{E}[E]$

To arrive at the cost metrics (the expected response times and rate of energy consumption) the recursions presented in Sections 6.1, 6.2, 6.3, and 6.4 must first be solved. An explicit step by step description of how to do so is given below.

1. Solve all repeated and boundary probability values. First, use equations (6.5) and (6.6). Note that (6.5) is a quadratic, therefore choose the root which is a valid probability, i.e. is between 0 and 1. Secondly, depending on the policy which is being analysed, use either (6.12), (6.21), or (6.27) to determine the boundary values for each row. Furthermore, for the top row, set all values of $P_C(C,j)$ for j between the boundary and repeating columns to 1. Once this step is complete, all remaining unsolved transition probability values will lie between the repeated and boundary columns of their corresponding rows.

2. Solve the remaining transition probabilities $P_{i'}(i,j)$. This is done by noting that $P_{i'}(i,j)$ does not depend on $P_{i'}(n,m)$ if $i > n$ or if $m < j$. To do this, start with the highest unsolved values of i and j and corresponding lowest feasible value of i'. For most decision variable choices, this is $i = i' = C - 1$ and j corresponding to one column left of the repeating portion of the chain. Then use either (6.13) and (6.17), (6.22), or (6.28) and (6.29) to solve the value. Repeat this process by decreasing j and moving left along the row. Do this for all i' such that $i \leq i' \leq C$. This will fully solve all probability transition values for row i. To find all probability values of interest repeat the above process for each row in a decreasing manner. After this step is complete, all transition probabilities will be explicitly determined.

3. Solve for the transition time values. Firstly, use (6.7) to get the repeating time values. Then solve for all non-repeating $T_{i,j}$ values. To do this, again the analysis exploits that $T_{i,j}$ does not depend on $T_{n,m}$ if $n < i$ or if $m < j$. Therefore, start by setting i and j to the highest unknown values and work down the row by firstly decreasing j, and then i while using the corresponding equations (6.14) and (6.18), (6.24), or (6.30).

4. Now all that remains is to solve for the holding cost values, and the energy consumption values, i.e. $H_{i,j}$ and $E_{i,j}$. The values for i and j are iterated through in the exact manner of the previous step, with the exception that the equations employed are now (6.15), (6.16), (6.19), (6.20), (6.25), (6.26), (6.31), and (6.32). Once this step is complete all of the recursions have been solved for.

With the recursion complete, one can now solve for the system metrics $\mathbb{E}[N]$ and $\mathbb{E}[E]$. To arrive at the expected number of jobs in the system, it is enough to know

the expected incurred holding cost over a single renewal cycle (denoted by $\mathcal{H}$), and the expected time to complete that same renewal cycle (denoted by $\mathcal{T}$). That is, from the renewal reward theorem,

$$\mathbb{E}[N] = \frac{\mathcal{H}}{\mathcal{T}}.$$

Furthermore, it is noted that $\mathcal{H}$ and $\mathcal{T}$ are easily derived from the transition costs determined above. By letting $(C_S, 0)$ be the cycle reference state, one can determine the total holding cost over a single cycle by determining the holding cost incurred before transitioning to state $(C_S, 1)$, and then the total holding cost incurred before returning to state $(C_S, 0)$ from $(C_S, 1)$. However, the latter value is $H_{C_S,1}$ by definition, and the holding cost incurred in state $(C_S, 0)$ before moving to state $(C_S, 1)$ equals 0 as there are no jobs present in the system. Therefore,

$$\mathcal{H} = H_{C_S,0}.$$

The above argument can also be applied to determine $\mathcal{T}$. That is, the expected time to transition to state $(C_S, 1)$ from state $(C_S, 0)$ is $1/\lambda$, plus the expected time to transition back to state $(C_S, 0)$ from state $(C_S, 1)$. Therefore,

$$\mathcal{T} = \frac{1}{\lambda} + T_{C_S,1}.$$

Similarly, the expected rate of energy consumption can also be determined. Letting $\mathcal{E}$ denote the expected amount of energy used over one cycle,

$$\mathbb{E}[E] = \frac{\mathcal{E}}{\mathcal{T}} \quad \text{where} \quad \mathcal{E} = \frac{C_S}{\lambda} E_{\text{Idle}} + E_{C_S,1}.$$

Leveraging the transition costs as such allows one to perform exact analysis on the expected response time, and expected energy consumption rate. In turn, this allows one to inspect the trade off between performance and energy efficiency.

6.6 Numerical Results and Observations

The numerical experiments are organized as follows. Each of the three policies (*bulk setup*, *dual threshold*, and *staggered threshold*) are evaluated under the same set of parameter configurations. The total number of servers (C) equals one of 20, 50, or 100. The setup rate (γ) equals one of 0.1, 0.01, or 0.001. The arrival rate (λ) is fixed to equal $C/2$, and the service rate (μ) is fixed at 1. Therefore, for the set of static servers to be stable on their own (without extra servers needing to turn on), it must hold that $C_S > C/2$. For experiments regarding the expected energy consumption rate, it is assumed that while idle a server accumulates cost at rate 0.7 and while in setup it accumulates cost at rate 1. This choice is influenced by the work presented in [5]. This range of parameters gives nine configurations for each of the policies. For each of these configurations $\mathbb{E}[R]$ and $\mathbb{E}[E]$ are evaluated with decision variables $k = 1, 3, 5, 7, 10, 15, 20, 25, 30$ and all valid values of C_S, i.e. $C_S = 0, .., C$. The experiments yield exact results and were done using standard Matlab libraries. While the Matlab code was not written with public use in mind, all source code needed to run these experiments can be found at [1]. The following three subsections, i.e. subsections 6.6.1, 6.6.2 and 6.6.3, are organized such that the discussions of the numerical experiments are at the beginning of each subsection, while all figures pertaining to these discussions can be found at the end of each subsection.

6.6.1 Bulk Setup

The behaviour of $\mathbb{E}[R]$ under the bulk setup policy is the first to be examined. This behaviour can be seen in Figures 6.3, 6.5, and 6.7. As expected, $\mathbb{E}[R]$ is monotonically decreasing in C_S. However, $\mathbb{E}[R]$ has a more interesting relationship with regards to the choice of k. One would perhaps expect that the lower the value of k the lower the expected response time would be. This is a reasonable thought since a lower value of k means a more proactive system, where servers are more inclined to turn on if there are jobs waiting. However, this is not always the case. Figure 6.7-(c) is a good example of this. Here for some lower values of C_S the expected response time for $k = 1$ is actually the largest among all curves shown. While at first perplexing, there is an intuitive explanation. While it is true that for a larger value of k the first few jobs to arrive and wait in the queue will have a longer response time, this is overcome by the fact that when the server turns on, there are now more jobs to process. Because there are more jobs to process, it will take longer for the server to become idle. Due to there being a larger window for a job to arrive when the server is already on, a larger value of k can actually result in a lower expected response time.

Observation 6.1. *There exist system configurations where increasing the value of k decreases $\mathbb{E}[R]$.*

Looking at the curves with larger values of k, i.e. the graphs on the right hand side of Figures 6.3, 6.5, and 6.7, shows another interesting behaviour. It seems that when k is sufficiently large, the expected response time decreases linearly with C_S until a point where it begins to converge to $1/\mu$. The point at which this changes in relation to C_S happens around $C/2$, but seems to vary slightly based on C. For example in Figure 6.3-(e) the relation seems to change further to the right of $C/2$, or in this case

10, at about the $C_S = 12$ region. The reason for $\mathbb{E}[R]$ converging to $1/\mu$ is clear. As the number of servers which are always on increases, the probability that the job has to wait in queue decreases, and its response time becomes its service time. On the other hand, if C_S is lower, the probability of a job having to wait in the queue increases. While it is not entirely clear why this increase in expected response time is linear, the following is noted. When a job arrives to the system and has to wait in queue, it can be served by one of two ways. Firstly, a fresh server can turn on and begin to process it. Secondly, a server which is currently processing a job can complete and begin to process the job that is waiting. Due to the bulk setup nature, the expected amount of time to turn on a single server increases with C_S. However, the expected time for a server to become available decreases with C_S. These two conflicting effects may counteract each other to produce a linear decrease in the expected response time of the system, in relation to C_S.

Observation 6.2. *For a large enough k, $\mathbb{E}[R]$ and $\mathbb{E}[E]$ can be described by two linear components. However, the value of k required to invoke this behaviour in the $\mathbb{E}[R]$ curve is less than the corresponding value of k for the $\mathbb{E}[E]$ curve.*

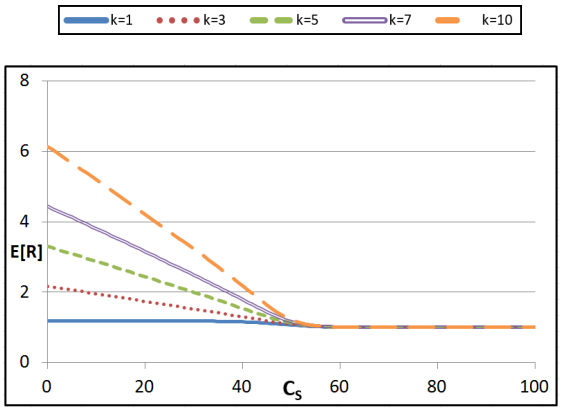
Focusing on the behaviour of the energy consumed by the system in Figures 6.4, 6.6, and 6.8 also leads to some interesting cases. Unlike the expected response time, the expected energy rate is not monotonically decreasing (or increasing) in C_S. This leads to local maxima and minima. Firstly, it is noted that around $\rho = \lambda/\mu = C/2$ there is a local maximum. The conjectured reason for this is the system is in a lose-lose scenario. That is, it is in a configuration where servers are regularly idling, while at the same time the system has a relatively high chance of being in a state where there are servers in setup. This is in contrast to the curve around $\rho + \sqrt{\rho}$, where there is a

local minimum, or in the case where γ is small, a global minimum. Here the system finds itself in a win-win configuration. That is, the chance of a job arriving to the system where there is not a server idling is low (consistent with the square root staffing rule [28]), and therefore the chance of servers being in setup is also low. On the other hand, servers which always remain on have a reasonable chance of being utilized, keeping the idling costs low. These two observations together make $C_S = \rho + \sqrt{\rho}$ an appealing choice, especially for systems with longer expected setup times.

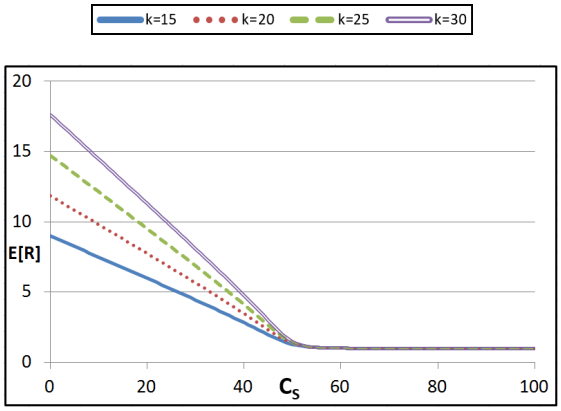
Observation 6.3. *For lower values of γ, $\mathbb{E}[E]$ has a local maximum around $C_S = \rho$.*

Looking back at the expected response time, the observation of $C_S = C/2 + \sqrt{C/2}$ being a good choice for the expected energy cost also holds from the performance stand point. The previous point that a job will rarely wait implies that the expected response time is close to its lower bound of $1/\mu$. This can be seen in Figures 6.3, 6.5, and 6.7. Furthermore, while the expected energy rate is sensitive to some configurations around $C_S = C/2 + \sqrt{C/2}$, it is less sensitive to the right. In other words, around $C_S = C/2 + \sqrt{C/2}$, $\mathbb{E}[E]$ increases at a lower rate when C_S increases, than if C_S were to decrease. This is also good news from a performance stand point, as $\mathbb{E}[E]$ is monotonically decreasing in C_S. Therefore, if one wished to err on the side of caution they could set their choice of C_S to the right of the minimum value without being punished too harshly.

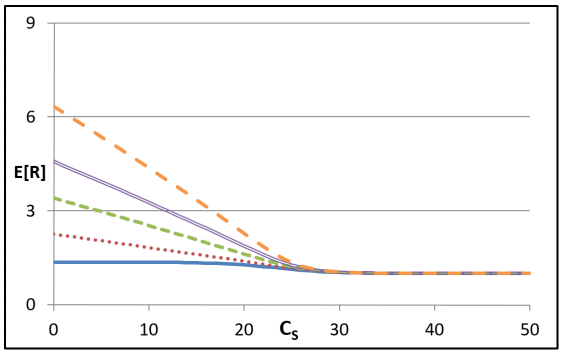
Observation 6.4. *For low values of γ (large expected setup times), the values for C_S corresponding to the minimums of $\mathbb{E}[E]$ and $\mathbb{E}[R]$ are approximately equal.*



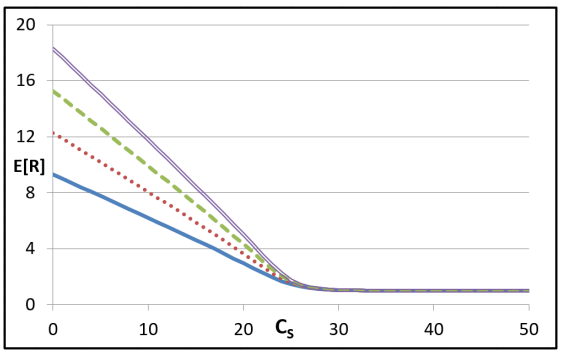
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



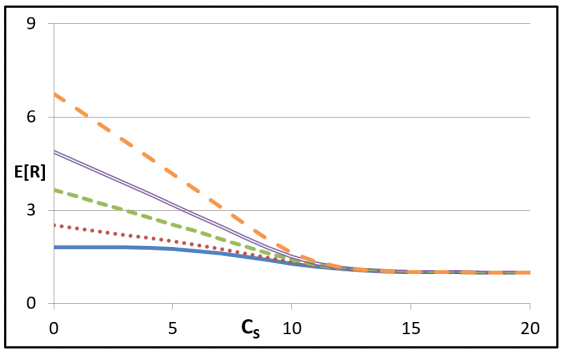
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



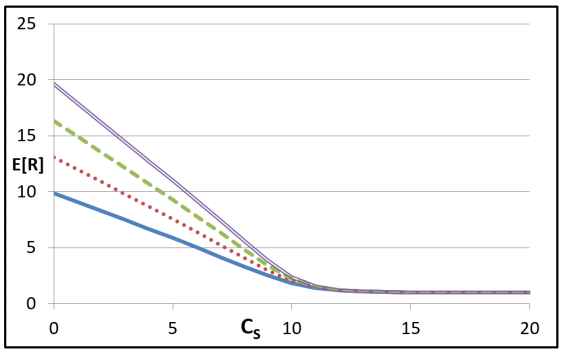
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

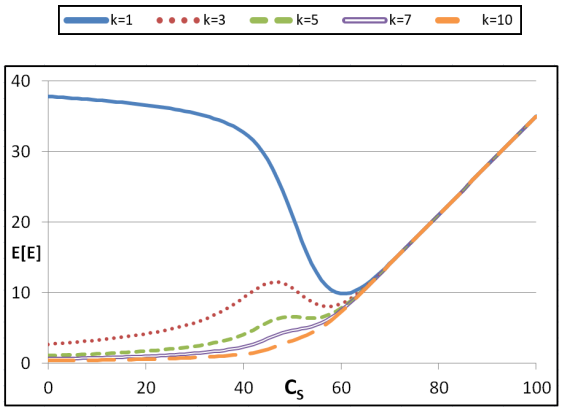


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

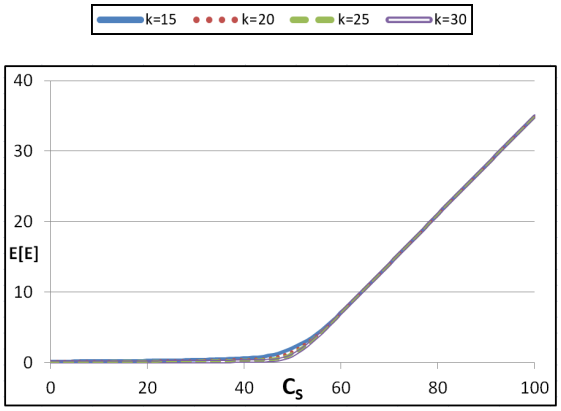


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

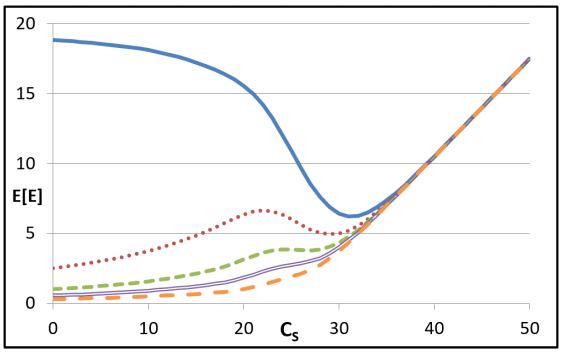
Figure 6.3: Expected response time vs C_S for $\gamma = 0.1$



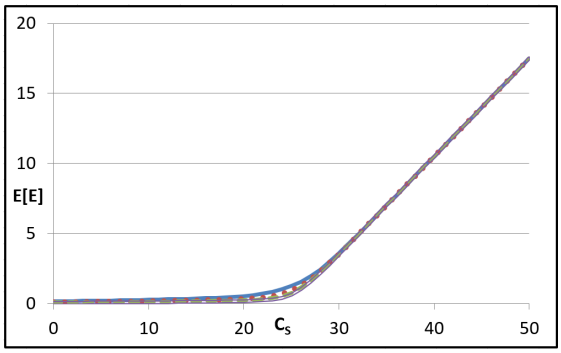
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



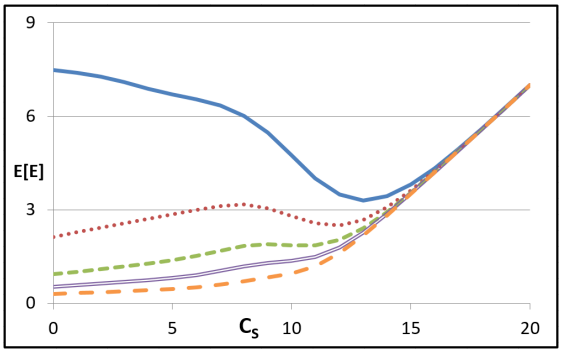
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



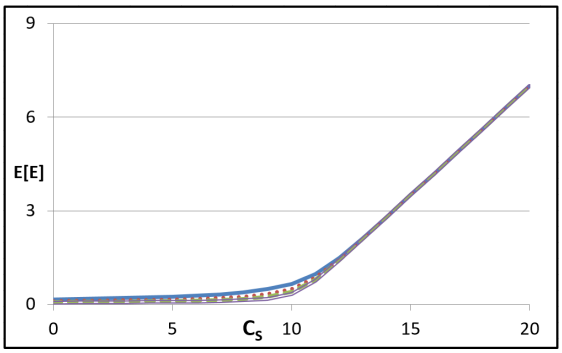
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

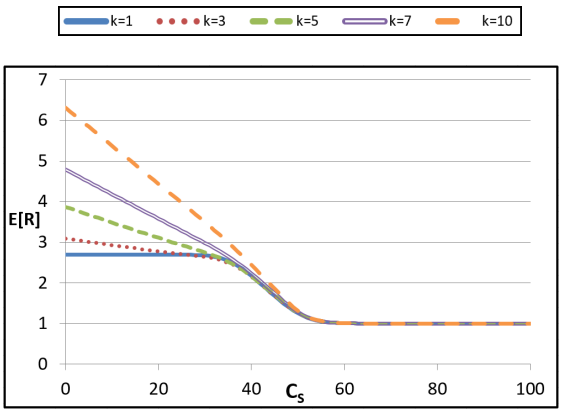


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

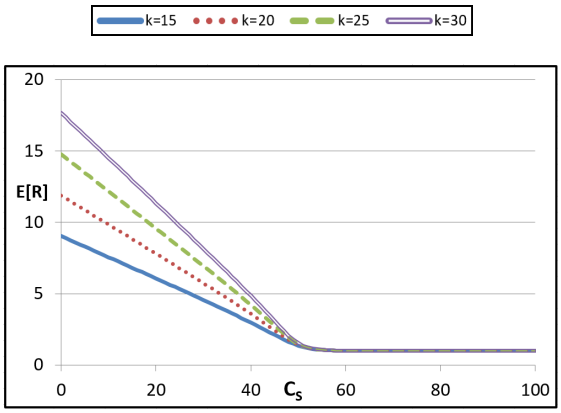


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

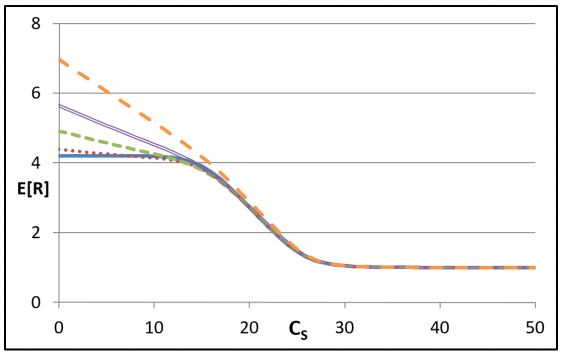
Figure 6.4: Expected energy consumption rate vs C_S for $\gamma = 0.1$



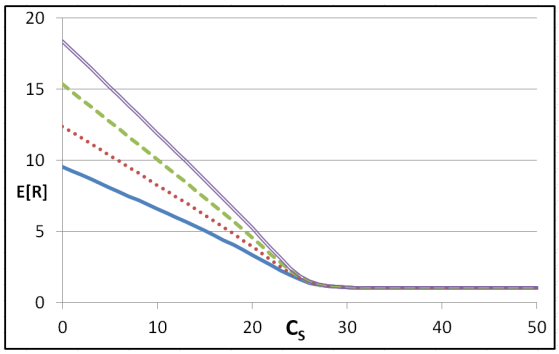
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



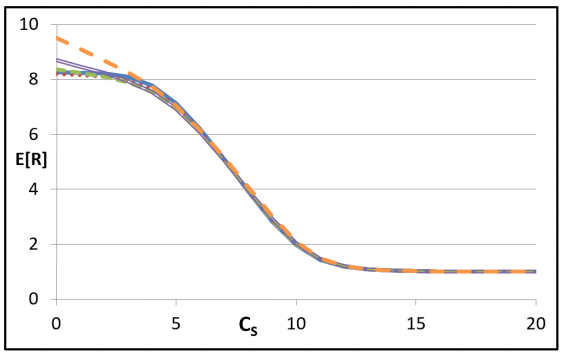
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



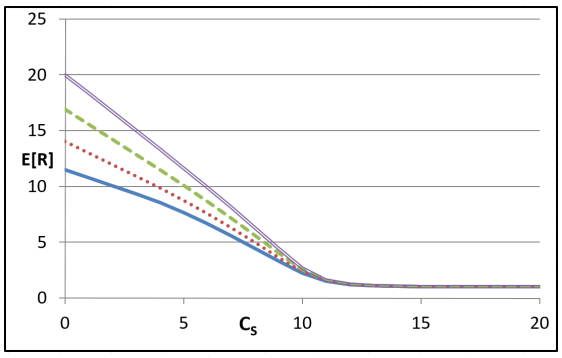
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

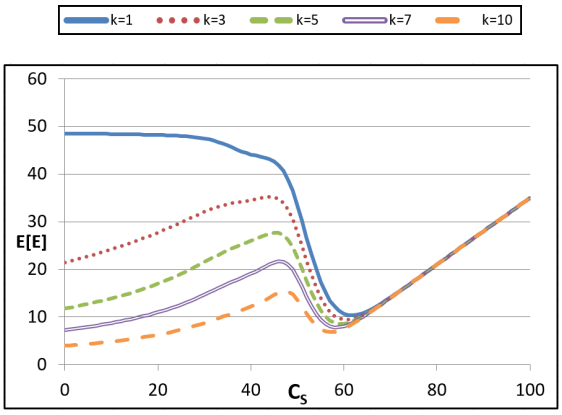


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

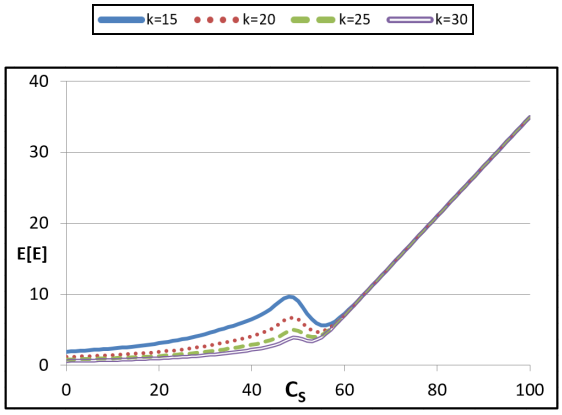


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

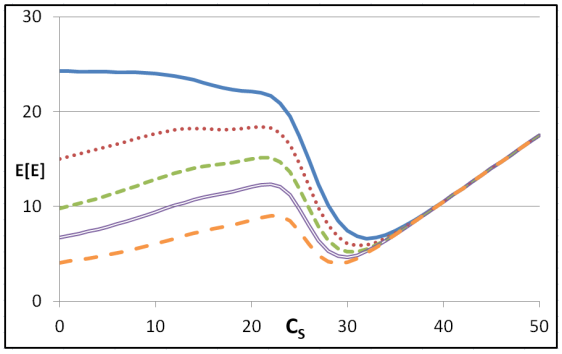
Figure 6.5: Expected response time vs C_S for $\gamma = 0.01$



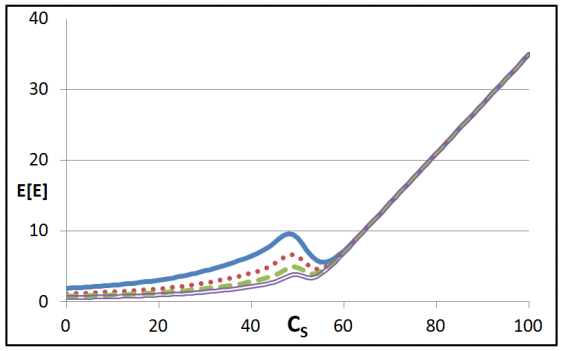
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



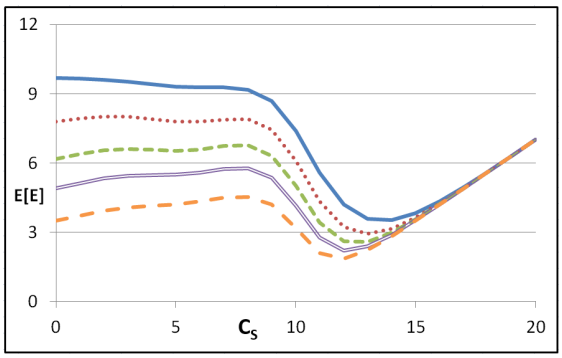
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



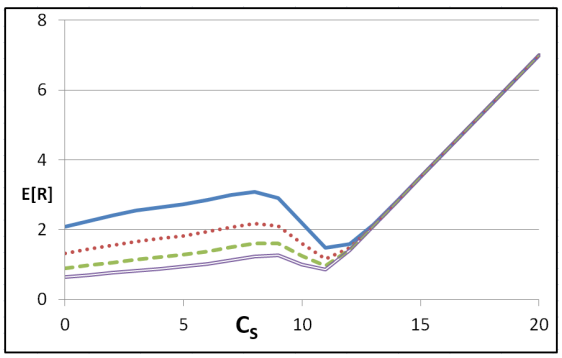
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

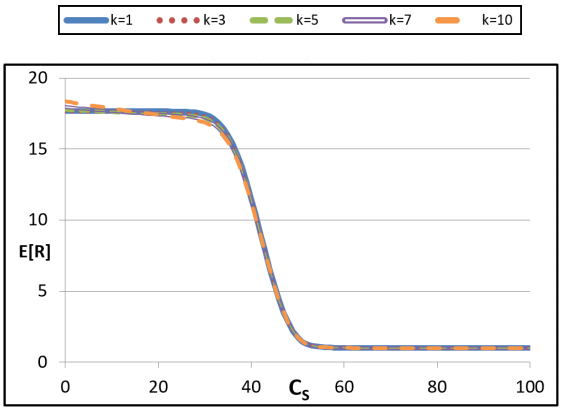


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

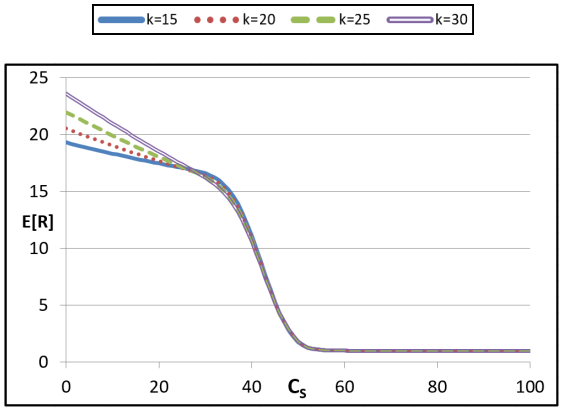


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

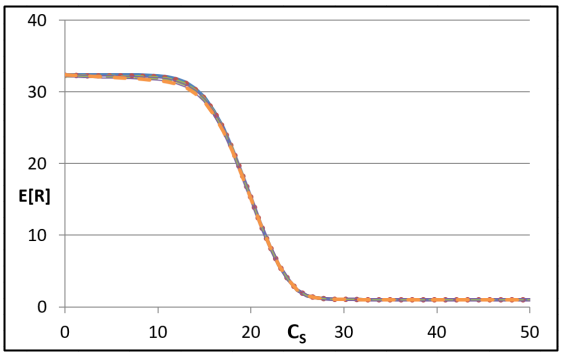
Figure 6.6: Expected energy consumption rate vs C_S for $\gamma = 0.01$



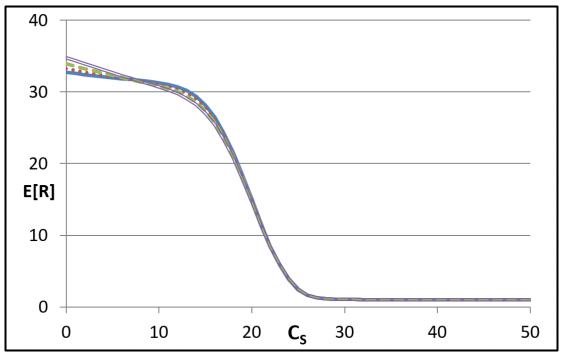
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



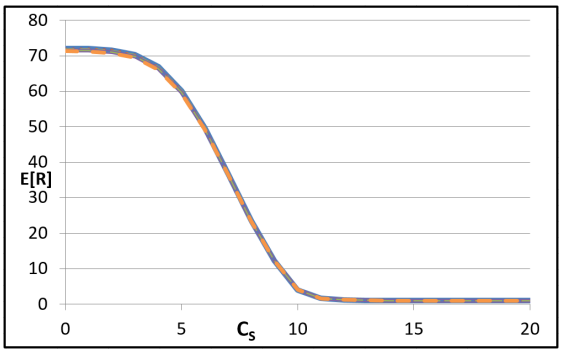
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



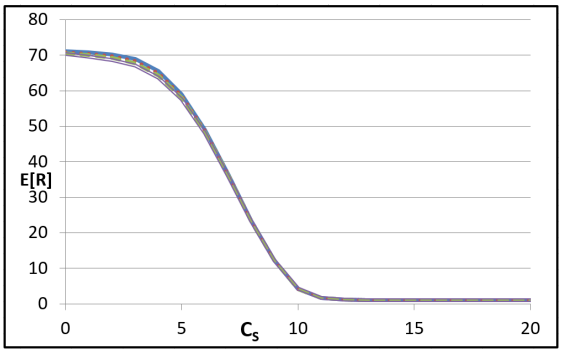
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k

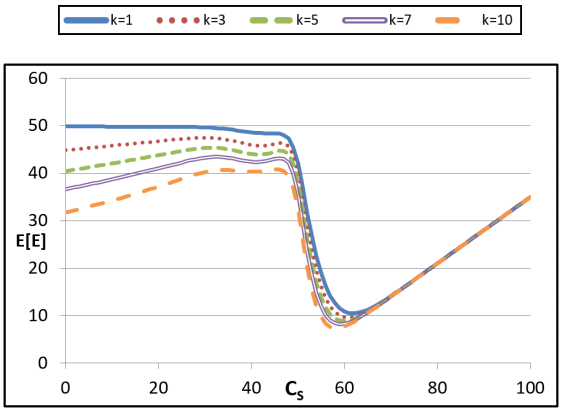


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$

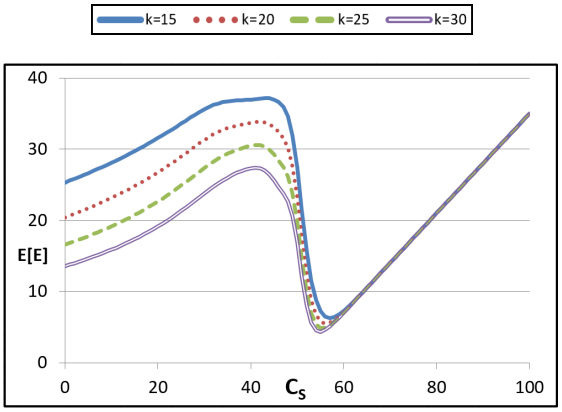


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

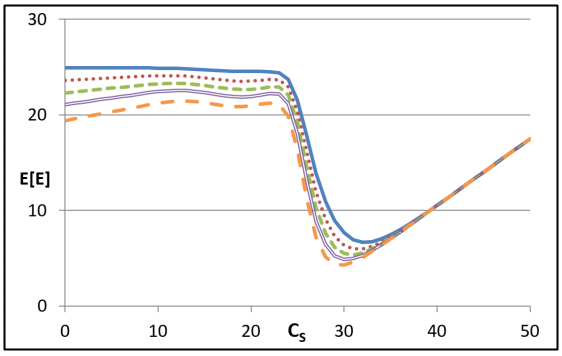
Figure 6.7: Expected response time vs C_S for $\gamma = 0.001$



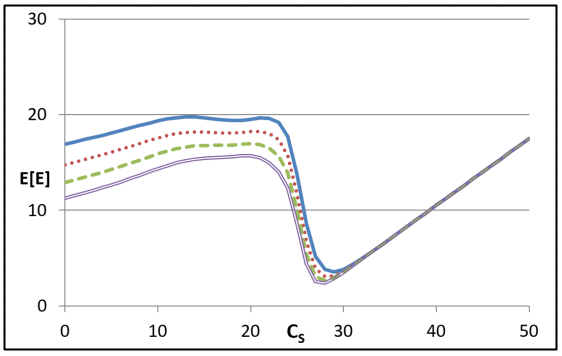
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



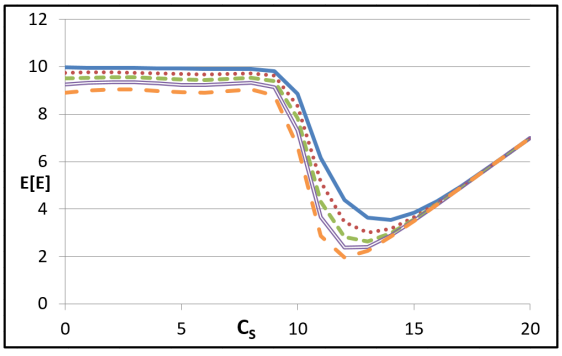
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



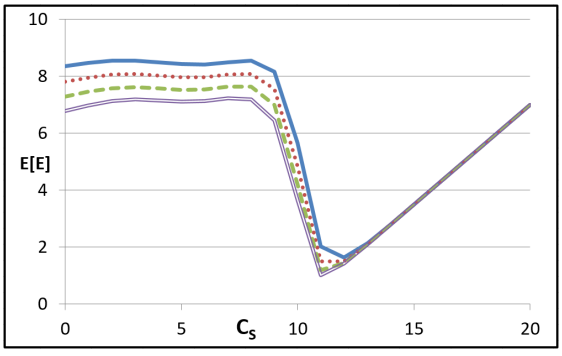
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k



(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$



(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

Figure 6.8: Expected energy consumption rate vs C_S for $\gamma = 0.001$

6.6.2 Dual Threshold

Examining the dual threshold policy leads to some interesting differences and similarities to the bulk setup policy. Firstly, the overall shape of the expected response time curves, i.e. the shape of the curves in Figures 6.9, 6.11, and 6.13, is similar to the corresponding curves for the bulk setup policy. This is due to the nature of these systems already having substantial constraints on how these curves must behave. Specifically there are two major criteria. Firstly, the expected response time must be monotonically decreasing in C_S. Secondly, the expected response time must converge to $1/\mu$. Within these constraints there is not a lot of interesting behaviour which can occur. Having said that, it is true that between the two policies there exist differences. For example, the response curves "bulge" due to the choice of k in Figure 6.11 while they remain closer to linear in Figure 6.5.

Observation 6.5. *The expected energy cost of the system is more sensitive to the choice of policy than the expected response time.*

Inspecting the energy curve tells a different story, i.e. Figures 6.10, 6.12, and 6.14 seem to be wildly different from the bulk setup curves. This is due to the quick to respond nature of the dual threshold policy. Consider the case where $C_S = 60$ and $k = 1$. In the bulk setup policy no servers would be in setup if there are sixty or less jobs in the system, and forty dynamic servers would be in setup or on if there were sixty-one or more jobs in the system. Examining how the dual threshold policy operates in this case tells a different story, for the same parameters. If there were forty or more jobs in the system, then the remaining forty servers would either be in setup or idle. This leads to a substantial difference in the expected energy cost as well as how one would choose the decision variables. Specifically, it would seem that the dual

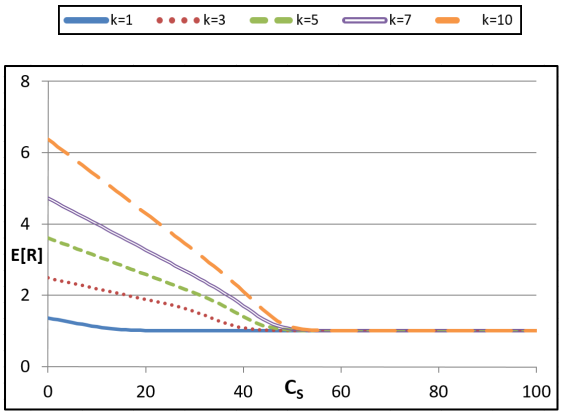
threshold policy over-provisions the system with servers which have a low probability of being turned off. For example, take Figure 6.12-(c), for the curve of $k = 1$ it can be seen that the minimum is around $C_S = 20$. The minimum of these energy curves is a sweet spot where not too many servers are turning on, nor are too many idling often. However, with these parameters it is clear that if the system only keeps twenty servers on, it will be unstable (until more are turned on). Therefore, if there are only twenty static servers, there should be a significant amount of setups and in turn a significant amount of energy costs. But this is not what one sees. This is because a certain number of the dynamic servers are behaving as static servers, i.e. a certain number of the remaining eighty servers which can be switched on or off are remaining on virtually all of the time. This should not be too surprising if one understands the nature of the policy. In this example, the twenty-first server will begin setup when there is at least one job in the system, and only turn off when the system is empty. However, the probability that the system is empty is extremely low, which will cause a server which is technically dynamic to behave instead as a static server. This effect continues (although to a lesser extreme) for the rest of the dynamic servers.

Observation 6.6. *For the dual threshold policy, a certain number of dynamic servers will behave as static servers. Furthermore, the lower the value of k, the larger this set of servers will be.*

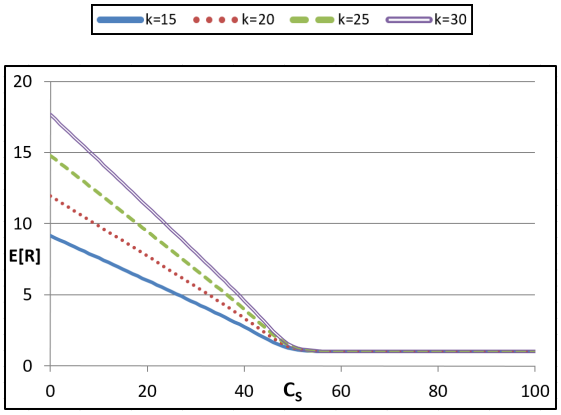
This observation can be further seen in the upper range of C_S in Figure 6.12-(c) for $k = 1$. One may expect (as was seen in other figures) the expected energy cost to increase linearly for higher values of C_S. This is due to noting that for higher C_S values these servers will be highly under utilized, and therefore each server will add

its idling cost to the overall energy cost. However, for $C_S > 75$ the expected energy cost is almost flat. This is again due to the fact that these servers which are now explicitly added to the set of static servers were in a sense implicitly already there.

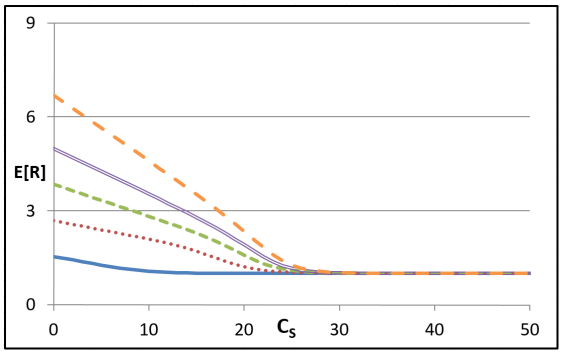
While the observation that the set of static servers is for all intents and purposes larger than C_S is an interesting one, it may be unappealing from an implementation standpoint. That is, it is harder to predict or determine how the system will actually behave. The decision variable C_S does not describe what it is meant to. Therefore, while the bulk setup policy has an unappealing setup criteria for implementation, it also has a more predictable behaviour with regards to the decision variables. This is the reason for the third policy analysed in this work - the staggered threshold policy attempts to incorporate the positive aspects of the bulk setup and dual threshold policies.



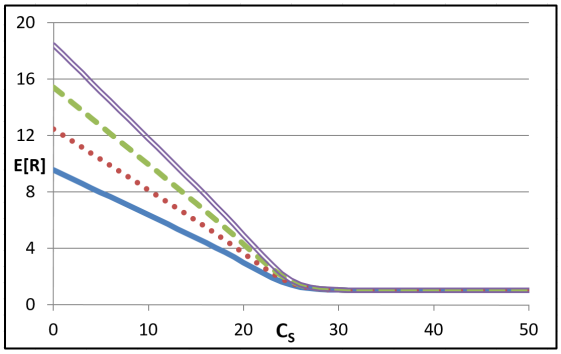
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



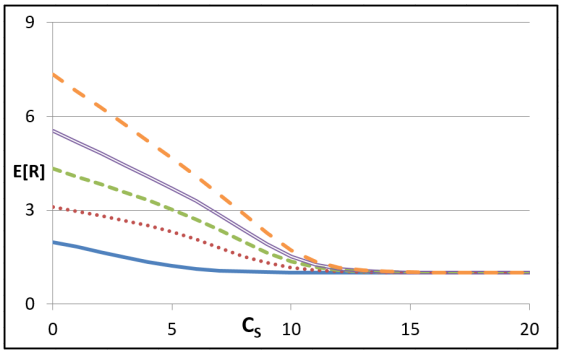
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



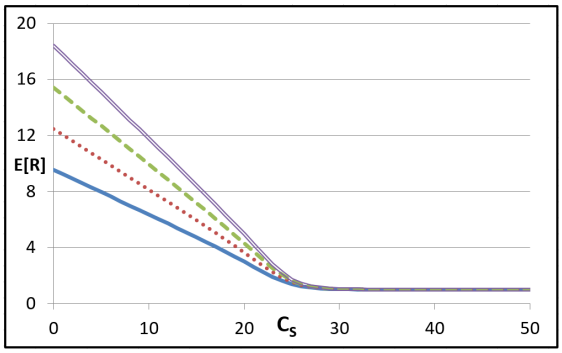
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

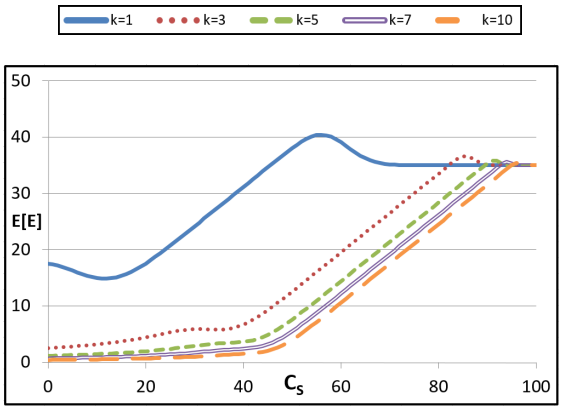


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

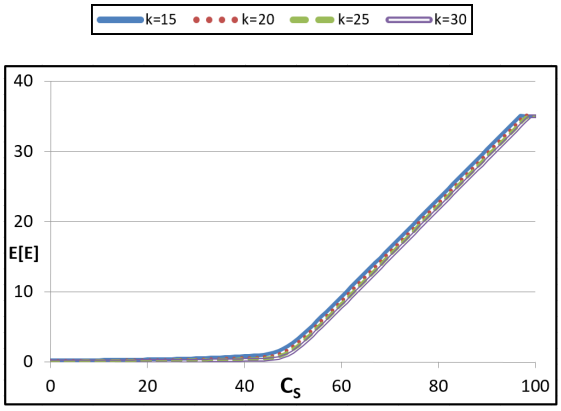


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

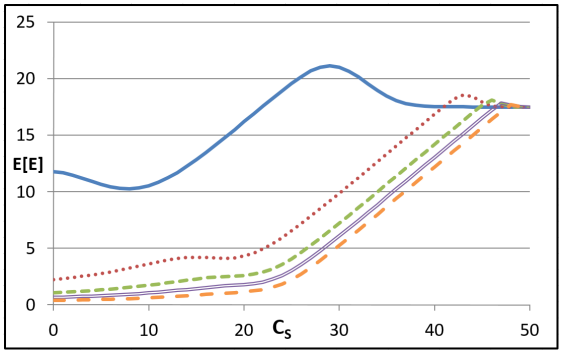
Figure 6.9: Expected response time vs C_S for $\gamma = 0.1$



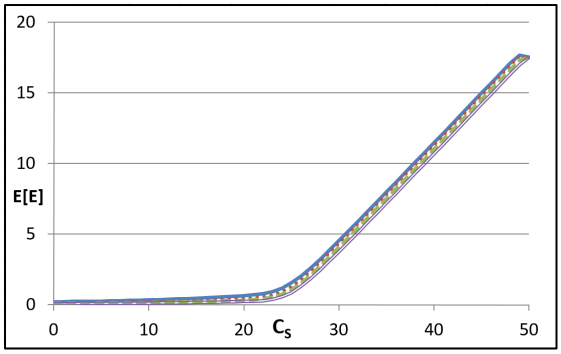
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



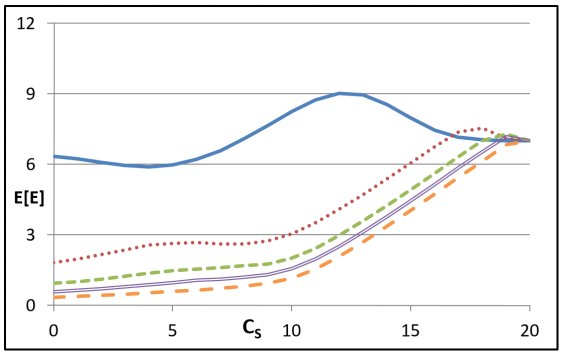
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



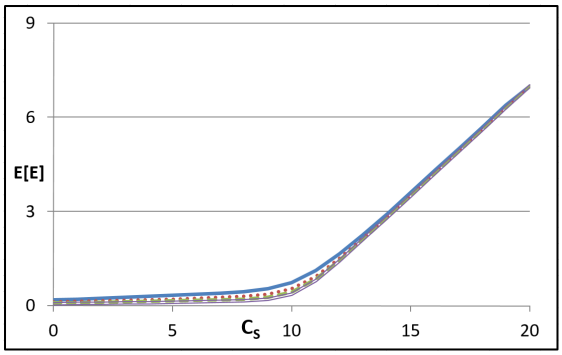
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

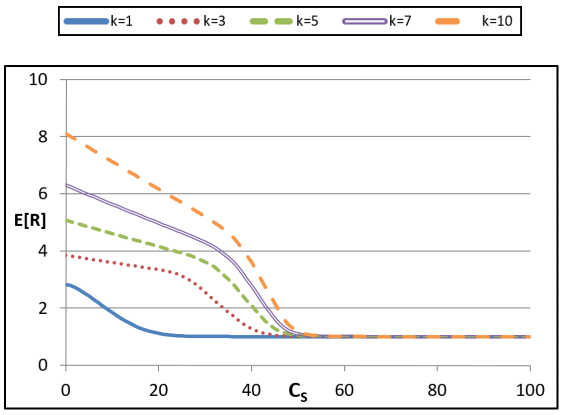


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

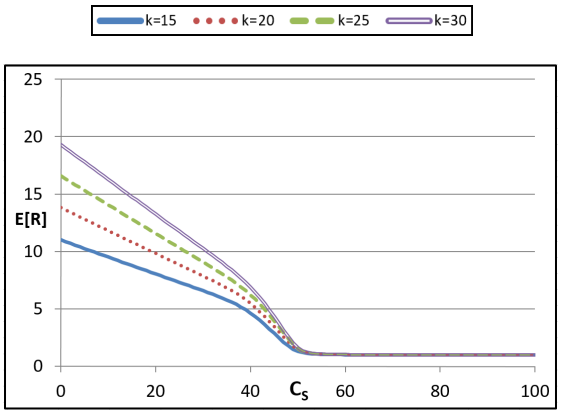


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

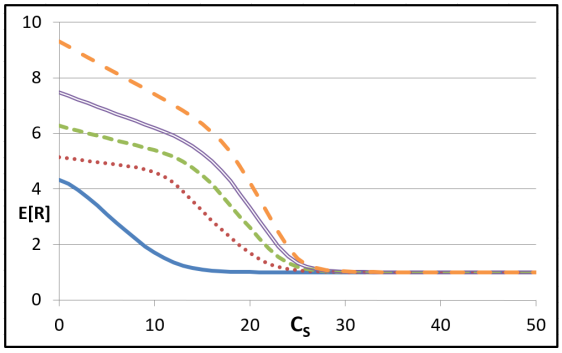
Figure 6.10: Expected energy consumption rate vs N^* for $\gamma = 0.1$



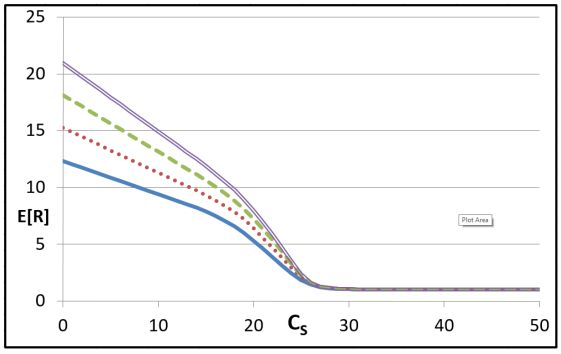
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



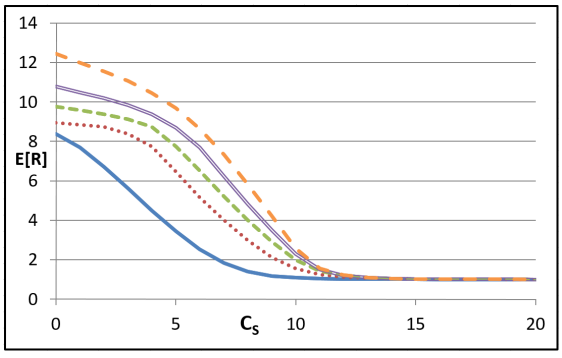
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



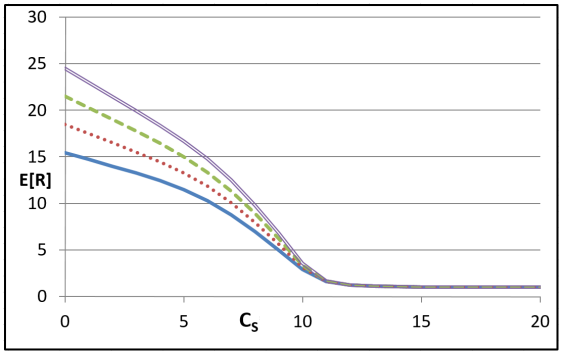
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

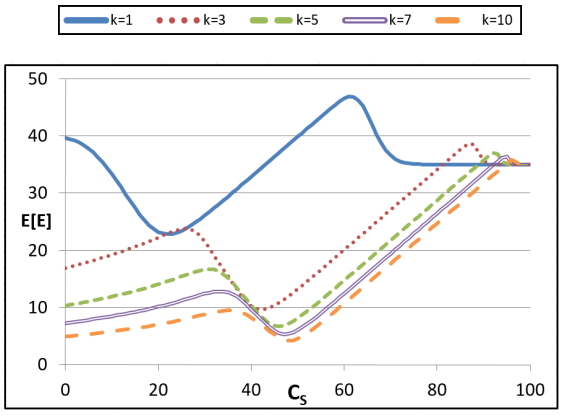


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

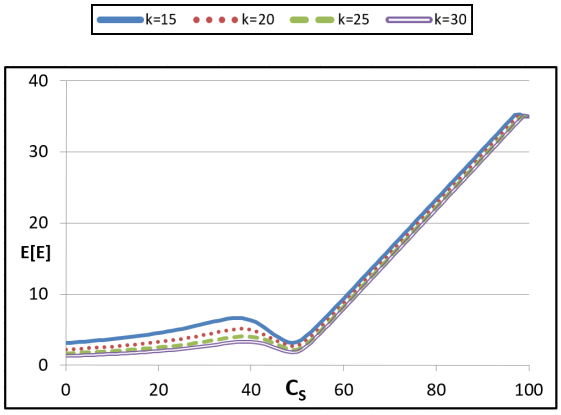


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

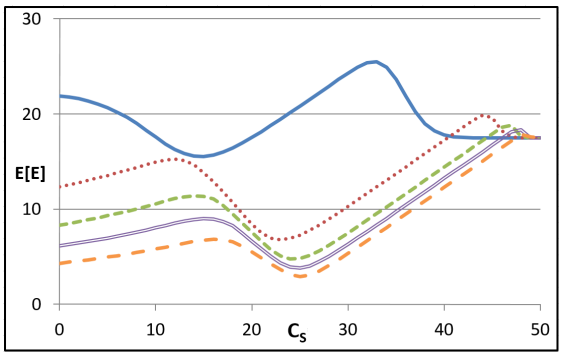
Figure 6.11: Expected response time vs C_S for $\gamma = 0.01$



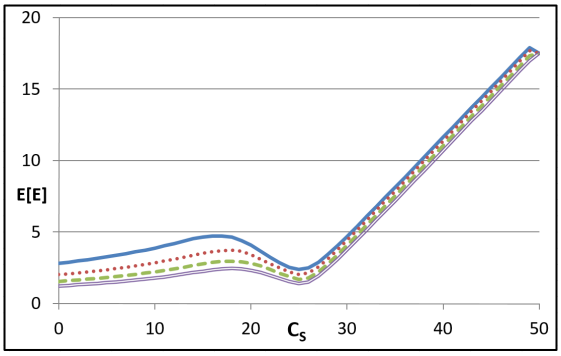
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



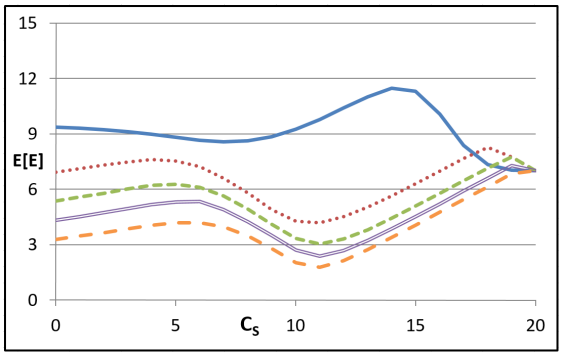
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



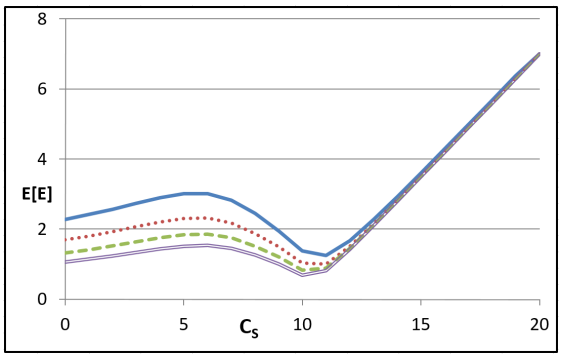
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

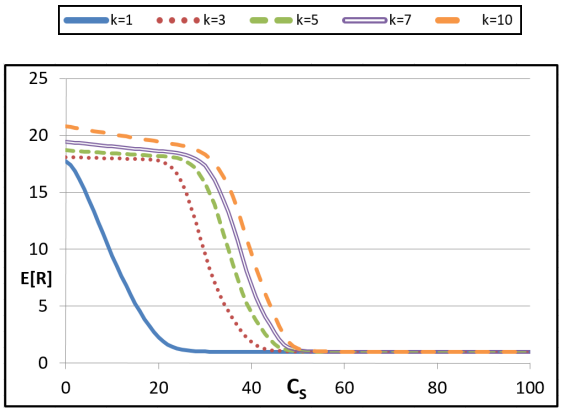


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

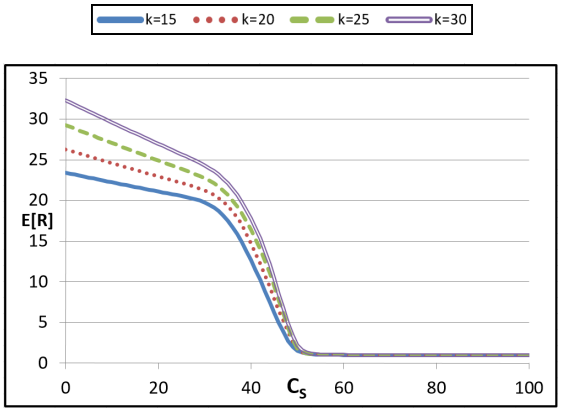


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

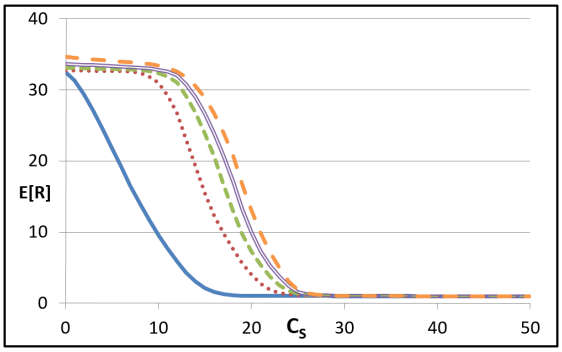
Figure 6.12: Expected energy consumption rate vs N^* for $\gamma = 0.01$



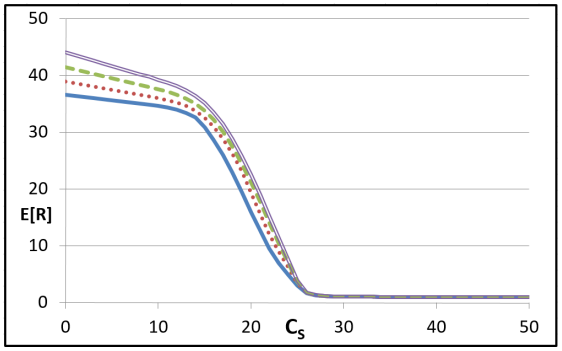
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



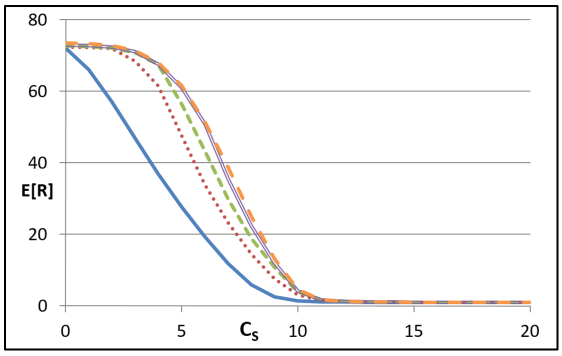
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



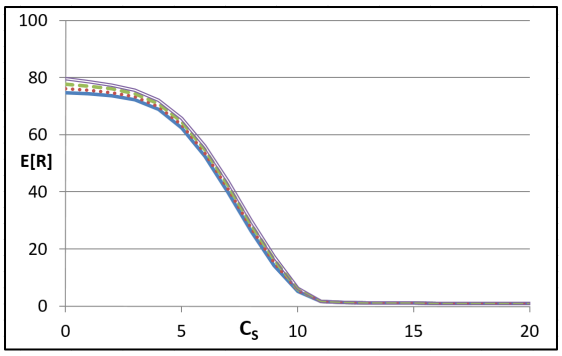
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k

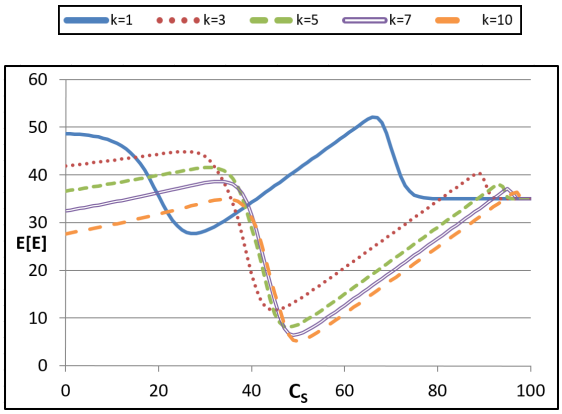


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$

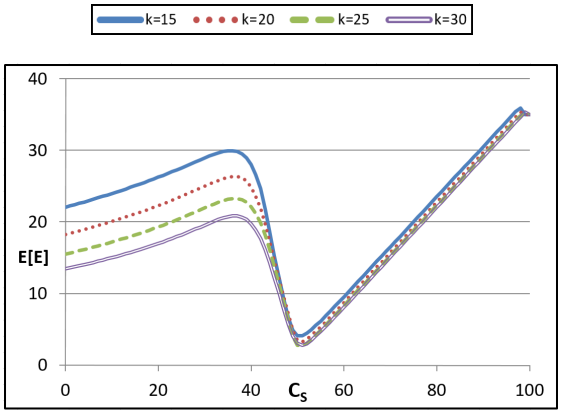


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

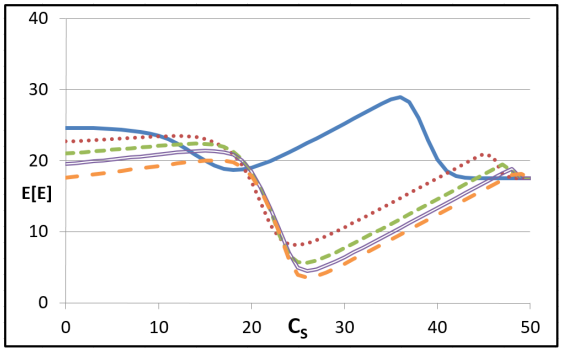
Figure 6.13: Expected response time vs C_S for $\gamma = 0.001$



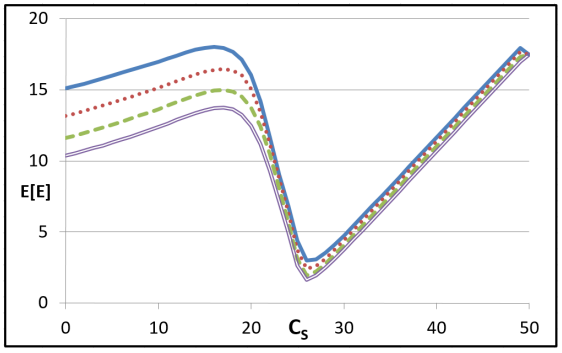
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



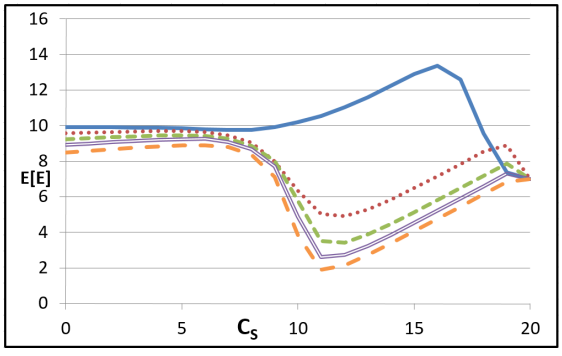
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



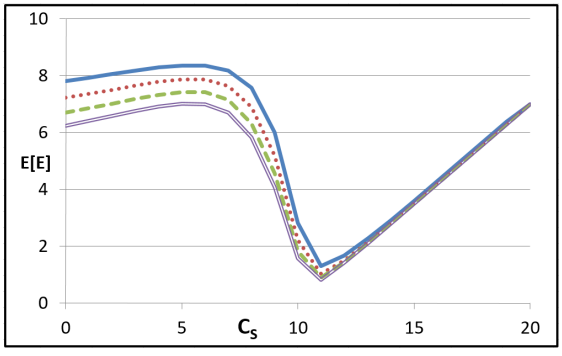
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k



(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$



(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

Figure 6.14: Expected energy consumption rate vs C_S for $\gamma = 0.001$

6.6.3 Staggered Threshold

The numerical results are completed with the *staggered threshold* policy. As discussed in Section 6.4, this policy aims to capture the predictability and stability of the bulk setup policy, while having a much more appealing implementation. The first thing of note is that in general these graphs look similar to those seen in Section 6.6.1. While it is true that both policies turn servers off when they idle, it should be obvious that the staggered nature of turning servers on makes the system slower to adapt to waiting jobs or bursts of traffic. However, the majority of the observations made for the bulk setup policy also hold here. One notable difference between these policies is that the response time does not decrease as close to linearly here as it did in the bulk setup results. Figure 6.17 is a good example of this, in contrast to Figure 6.5.

Observation 6.7. *The overall shape of the* $\mathbb{E}[R]$ *and* $\mathbb{E}[E]$ *curves with respect to the decision variables is relatively insensitive to using the bulk setup or staggered threshold policies.*

Arguably the most important similarity to that of the bulk setup policy is the presence of the aforementioned "sweet spot" in the energy curves. That is, the expected rate of energy consumption often has a minimum relatively close to $\rho + \sqrt{\rho}$, where $\rho = \lambda/\mu$, for many of the energy curves. It should be noted that for some of the energy curves with large values of k, such as Figure 6.16-(d), while the minimum is actually at C_S, the value at $\rho + \sqrt{\rho}$ is still only a slight increase in value of the minimum. Therefore, for all the experiments conducted, it holds that $\rho + \sqrt{\rho}$ is a reasonable choice for C_S with regards to energy costs. However, by inspection it is clear to see this is also a good choice for C_S from a performance standpoint. This will be shown later in this section. Moreover, inspecting the choice of k for this value of C_S leads to an

interesting implication.

Observation 6.8. *The expected energy costs for the bulk setup and staggered threshold policies are decreasing in k.*

Reviewing Figures 6.16, 6.18, and 6.20 one will note that for all fixed values of C_S the expected energy cost is decreasing in k. That is, the longer the system is willing to wait before turning servers on, the lower the energy costs will be. This is an intuitive result, but perhaps not obvious. Consider the following fallacious argument. If k is large, the system could be faced with a situation where there are a lot of jobs in the system by the time the next server turns on, which will cause a greater number of servers to be turned on in the short run. Due to this large number of servers now on, the system will quickly clear out all of the current jobs. Jobs departing from the system due to dynamic servers being turned on will now cause static servers to become idle where they otherwise may have been busy, thus incurring a higher expected energy cost. However, from the numerical results it can be seen that this is not the case (at least for the parameters examined). The reason the energy costs are lower for higher values of k is that dynamic servers are less likely to "thrash". For example, if a server begins its setup when there is one job waiting ($k = 1$), it will incur an initial setup cost in the short run that it may otherwise not for a larger value of k, but it may also quickly clear the job out, switch off, and then find itself in the same situation of one job waiting to be served in the near future. This causes multiple setup cycles to occur to deal with a set of jobs which a higher value of k may deal with using only a single setup or potentially without any setups at all. Due to a lower number of server setups for a higher value of k, the expected energy cost is strictly lower. Therefore, if energy costs are the only concern, one should choose the highest possible value of

k. But a higher value of k could have a (potentially disastrous) negative impact on performance. However, leveraging the previous observation for a reasonable choice for C_S, i.e. $C_S = \rho + \sqrt{\rho}$, this may not be the case. Viewing Figures 6.15, 6.17, and 6.19 one notes that around $C_S = \rho + \sqrt{\rho}$ the expected response time is quite insensitive to the choice of k. Therefore, the largest possible value of k should be chosen. Since there is no restriction on the ceiling of k, one should let $k \to \infty$. But if that is the case, the system degenerates to the well known $M/M/C_S$ queueing system where $C_S = \rho + \sqrt{\rho}$.

Observation 6.9. *For all parameter configurations examined here, for both the expected response time and expected energy costs, the degenerate solution of using an $M/M/C_S$ queue is near-optimal for some C_S around $\rho + \sqrt{\rho}$.*

While perhaps at first this is a disappointing result, since it implies energy costs cannot be saved, it gives an elegant and simple solution to what appears on the surface to be a complex problem. It is argued that for linear cost functions the bulk setup policy is a reasonable approximation of the optimal policy. However, the bulk setup turn on scheme (and in turn the bulk setup policy) hinges on interruptible setups and exponentially distributed setup times. Therefore, in turn, the staggered threshold policy is analysed. It is found that an $M/M/C_S$ queue is close to optimal for both of these policies. Thus, it is argued that an $M/M/C_S$ is close to optimal across all potential policies for some C_S. Again, this gives rise to a simple solution which is easy to implement.

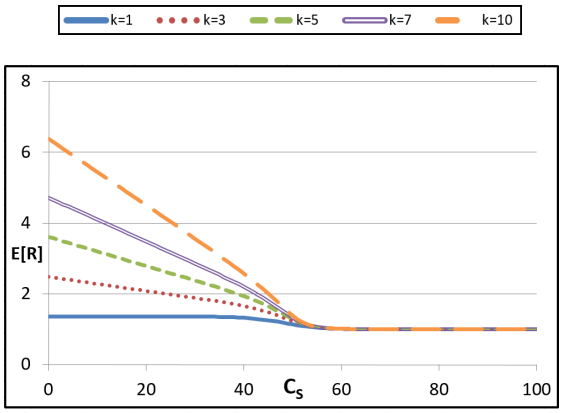
These results would suggest that near-optimal control of these multiserver systems can be achieved with a single decision variable, C_S. Moreover, the choice of C_S is

solely dependent on ρ. In other words, to have a near-optimal system, one need only concern themselves with accurately determining λ and μ (and not potentially complicated setup and turn off criteria). Such a solution offers an additional benefit. Researchers often choose to incorporate the expected *rate of switching* (how often servers turn on/off) to capture the wear-and-tear cost of the hardware [9, 46, 51]. It immediately follows that this cost metric is trivially minimized when only a static allocation of servers is employed (no switching occurs). Therefore, any well-formed cost function including the expected rate of switching also agrees with the degenerate solution.

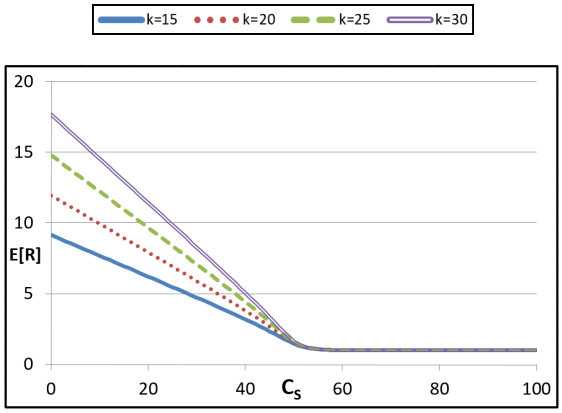
The argument of an $M/M/C_S$ queue being a near-optimal solution is further reinforced by revisiting Observation 6.8 in more detail. Observation 6.8 tells us that to minimize the expected energy cost, the best choice of k is the largest value of k, or $k = 30$ if limited to the choice of the experimental parameters. But if the system is stable, specifically if the system has approximately $\rho + \sqrt{\rho}$ static servers, what is the physical interpretation of such a large value for k? Clearly, the probability that there are greater than n jobs in the system for the model, is less than or equal to the probability that there are greater than n jobs in an $M/M/C_S$ queue. That is, $P(N > n) < P(N_{M/M/C_S} > n)$, where $N_{M/M/C_S}$ is a random variable denoting the number of jobs in an $M/M/C_S$ queue, and $C_S < C$. But using $C_S = \lceil \rho + \sqrt{\rho} \rceil = 58$ and $C = 100$, one can do a quick calculation to find that $P(N > 87) \approx 0.0023$. In other words, if $k = 30$, at least 434 jobs out of 435 will not cause the first dynamic server to begin its setup process when they arrive. Furthermore, approximately only 1 job out of every 44,000 has a chance of initiating the setup process of the second

dynamic server when it arrives. Therefore, the physical interpretation that larger values are a good choice for k corresponds to saying the system should not utilize its dynamic servers, but instead be statically provisioned.

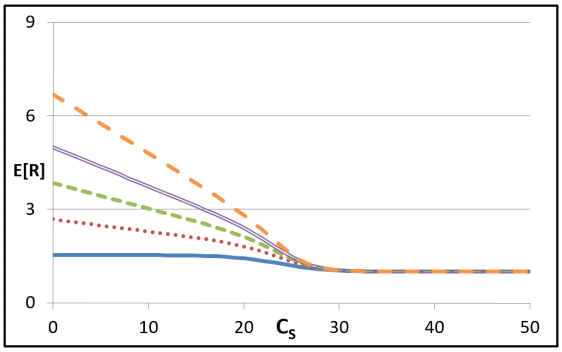
This simple solution, while derived from a mathematical model, agrees with practical results from real-world policies. As an example, in [22] the authors note that *dynamic capacity management policies* are too quick to turn idle servers off. Examining Figures 6.3-6.20 reinforces this notion via analysis. One can quickly observe that not having a reasonable number of static servers (say 40 and below for these particular experiments) results in a punishing lose-lose scenario.



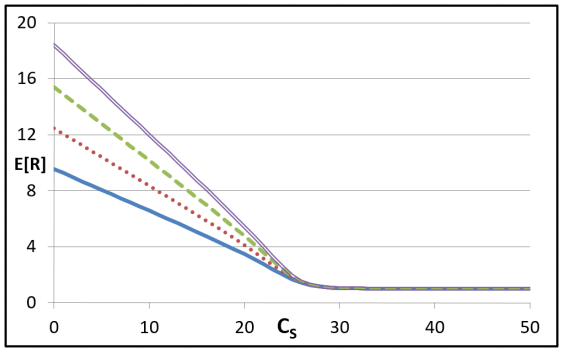
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



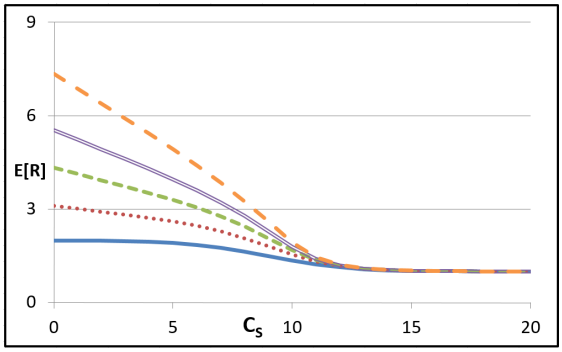
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



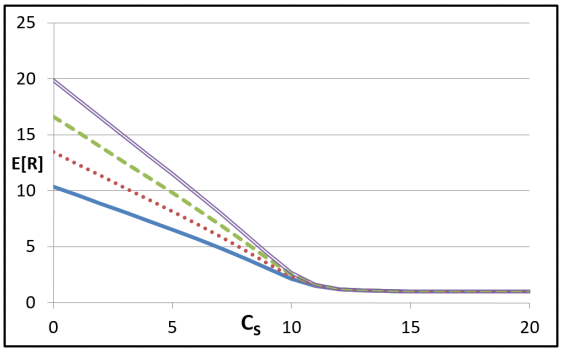
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

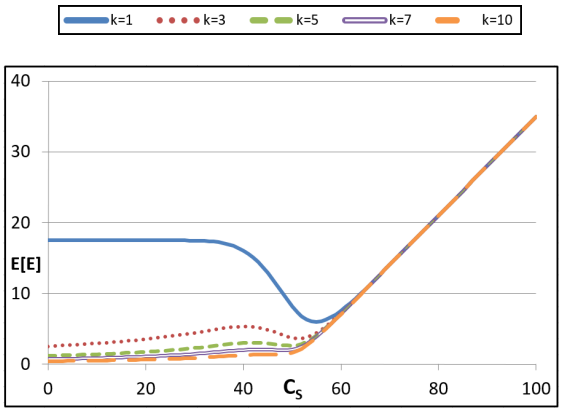


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

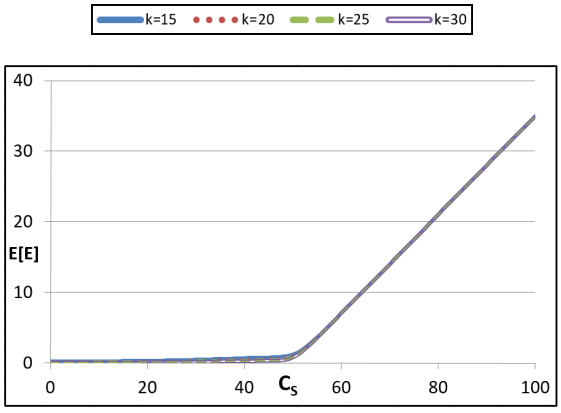


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

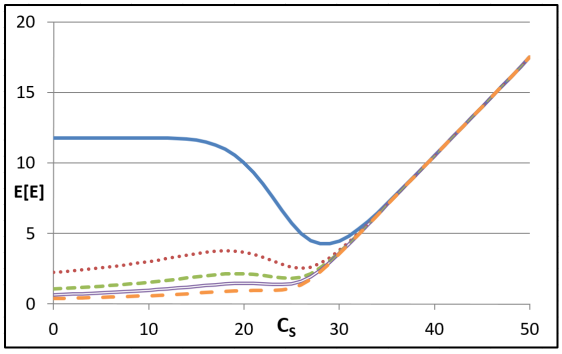
Figure 6.15: Expected response time vs C_S for $\gamma = 0.1$



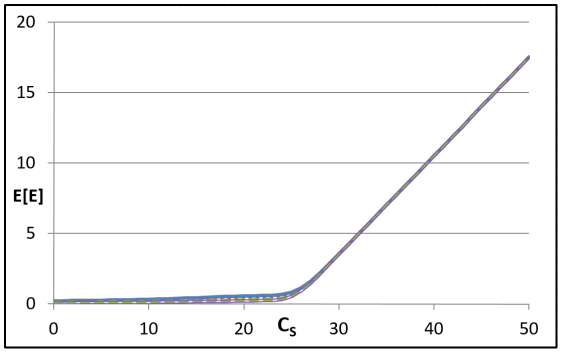
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$



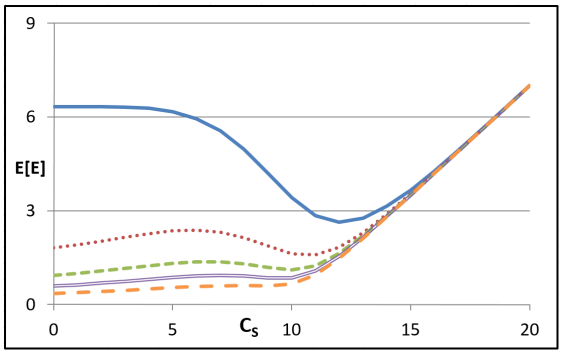
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.1$, with larger k



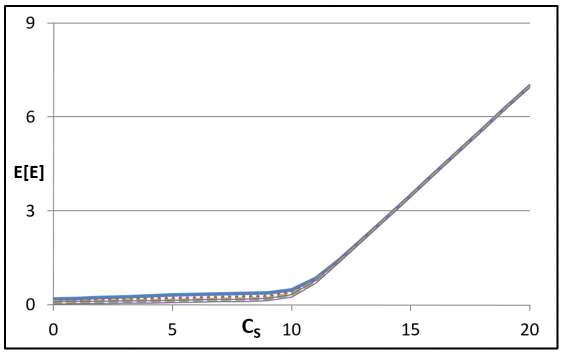
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.1$, with larger k

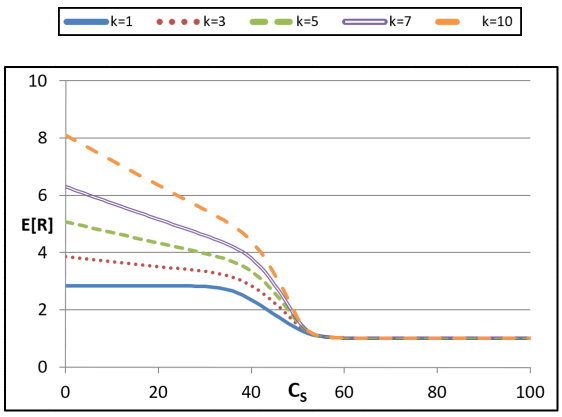


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$

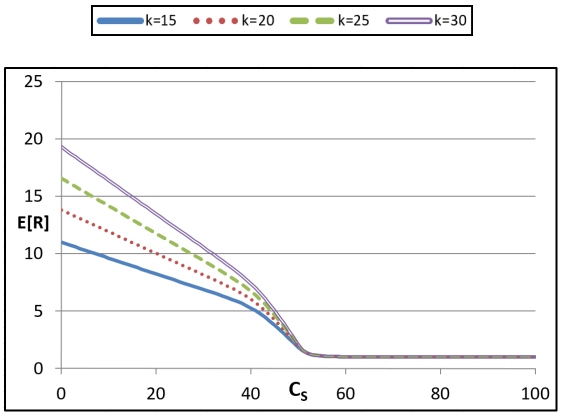


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.1$, with larger k

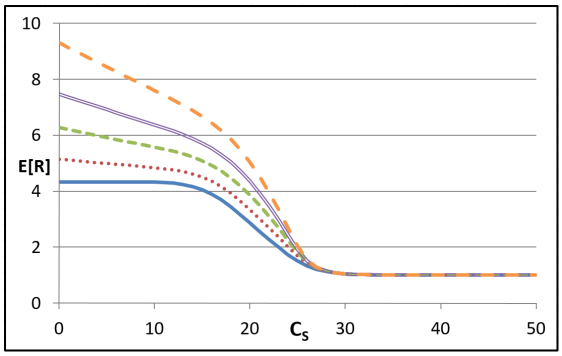
Figure 6.16: Expected energy consumption rate vs C_S for $\gamma = 0.1$



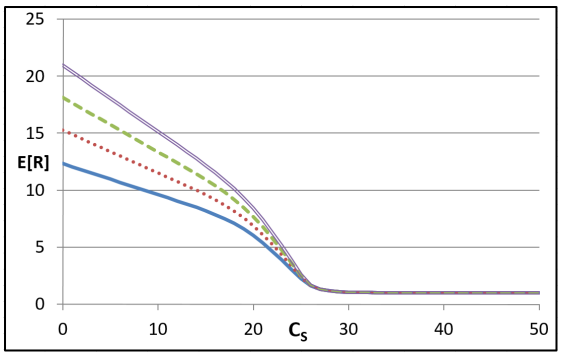
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



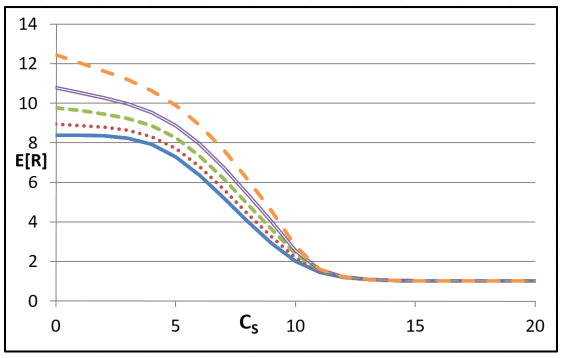
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



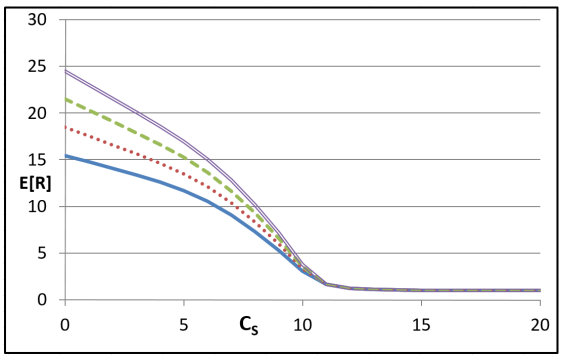
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

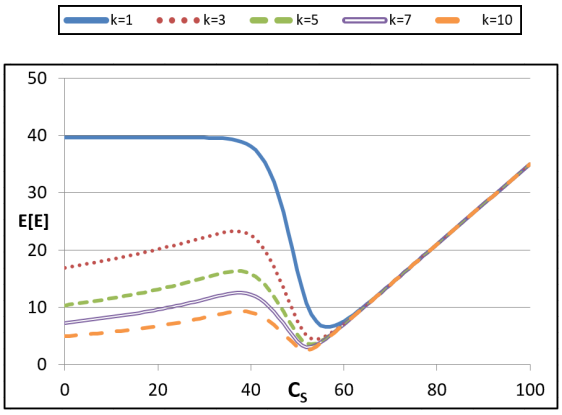


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

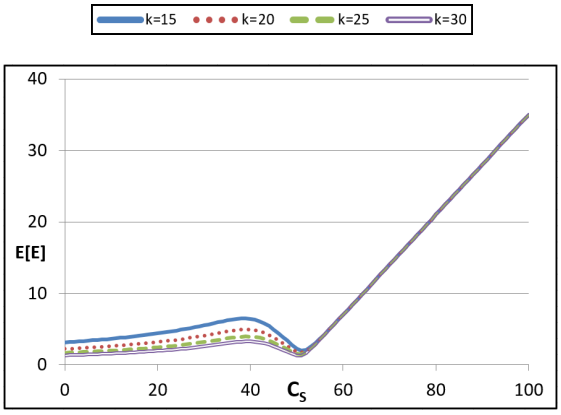


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

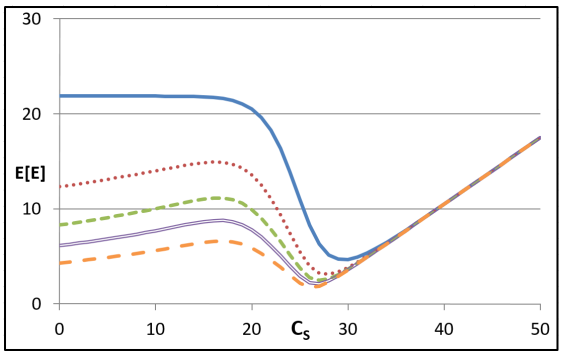
Figure 6.17: Expected response time vs C_S for $\gamma = 0.01$



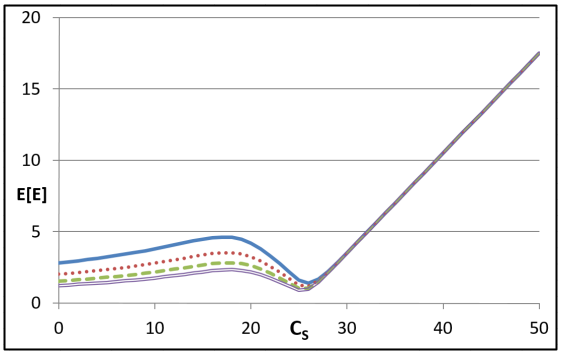
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$



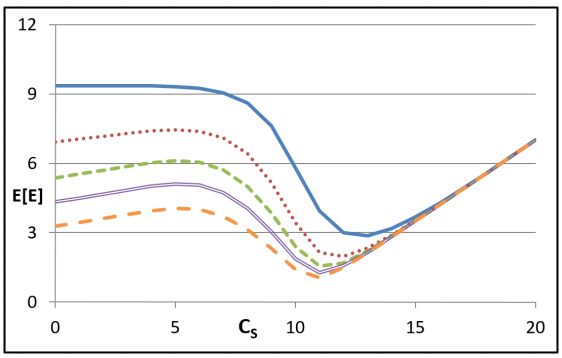
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.01$, with larger k



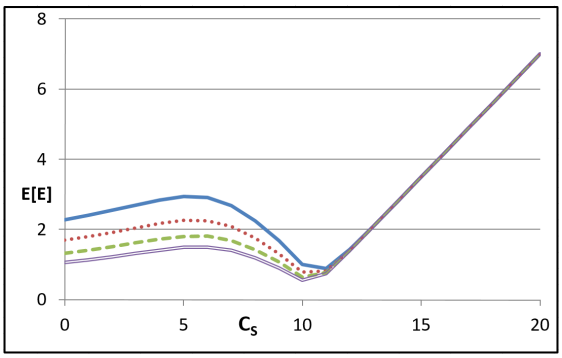
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.01$, with larger k

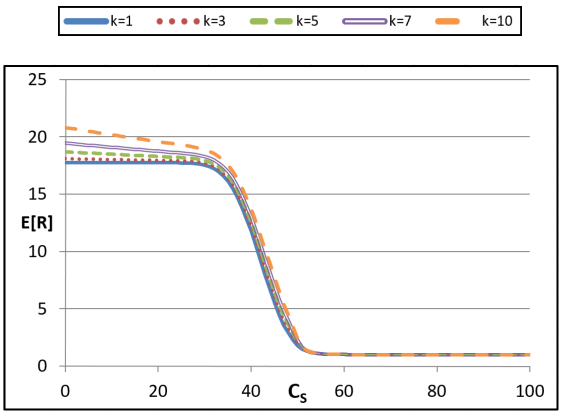


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$

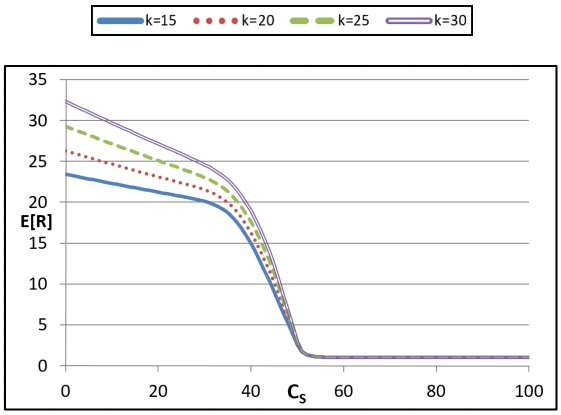


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.01$, with larger k

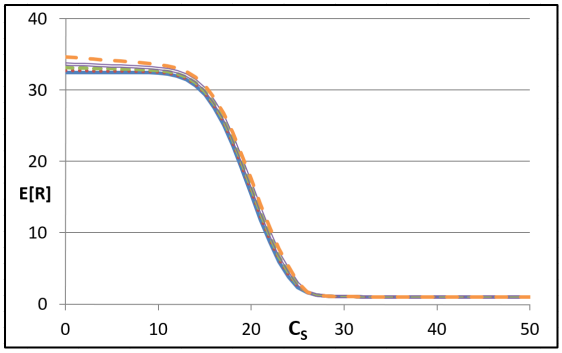
Figure 6.18: Expected energy consumption rate vs C_S for $\gamma = 0.01$



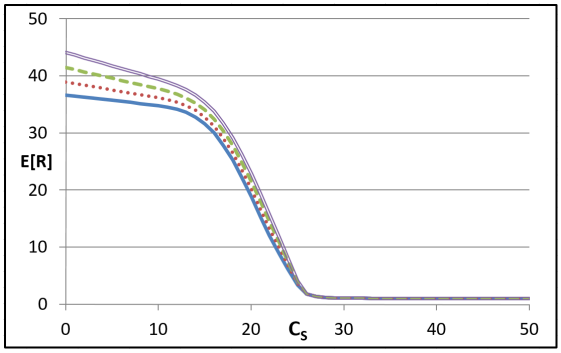
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



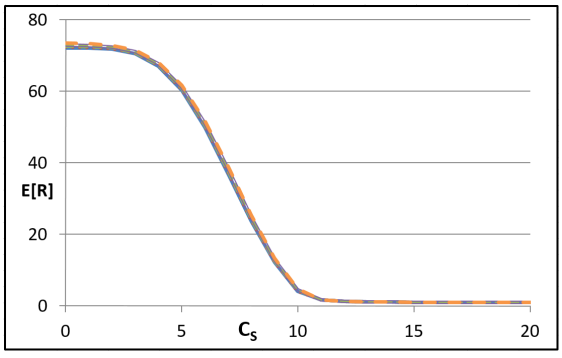
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



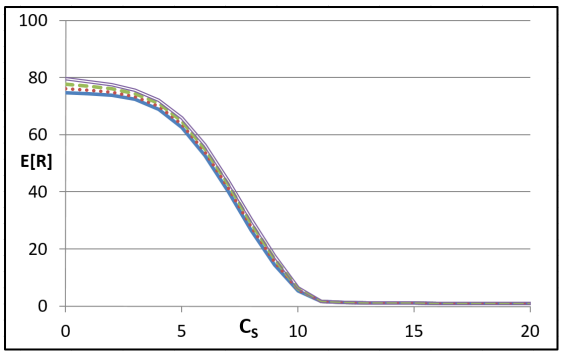
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k

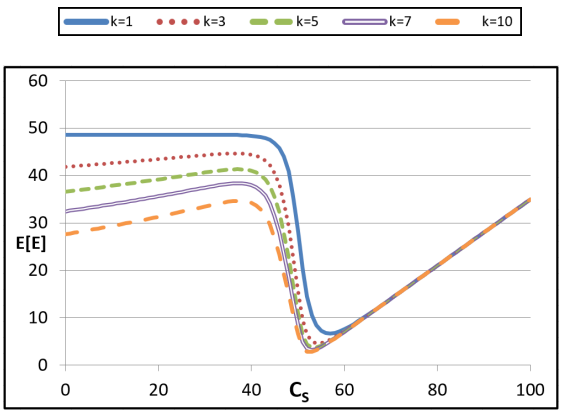


(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$

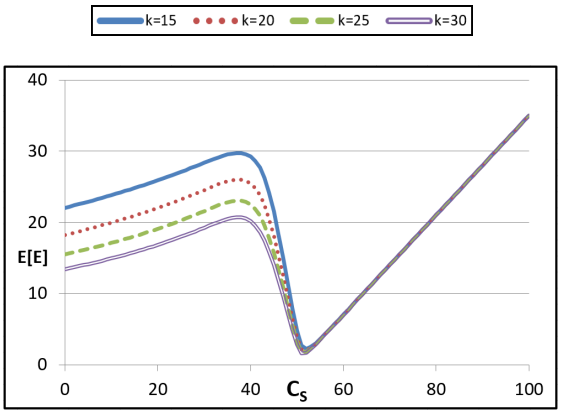


(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

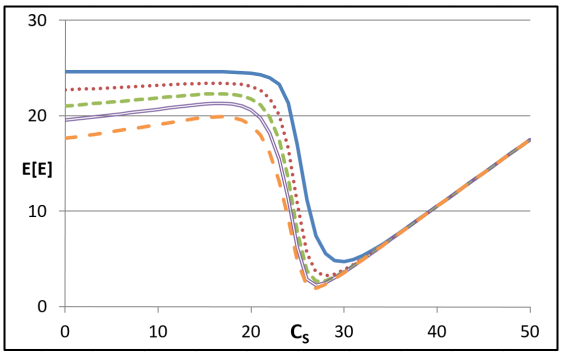
Figure 6.19: Expected response time vs C_S for $\gamma = 0.001$



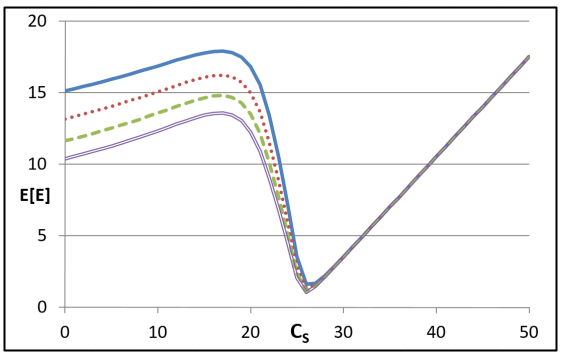
(c) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$



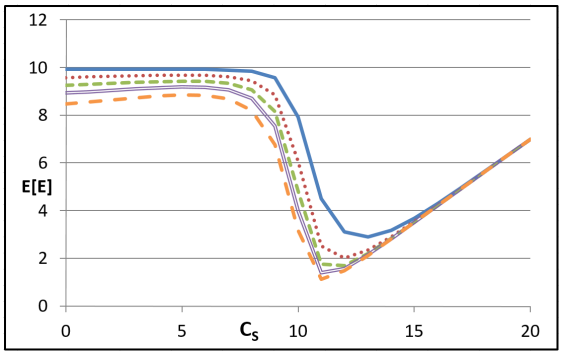
(d) $C = 100,\ \lambda = 50,\ \mu = 1,\ \gamma = 0.001$, with larger k



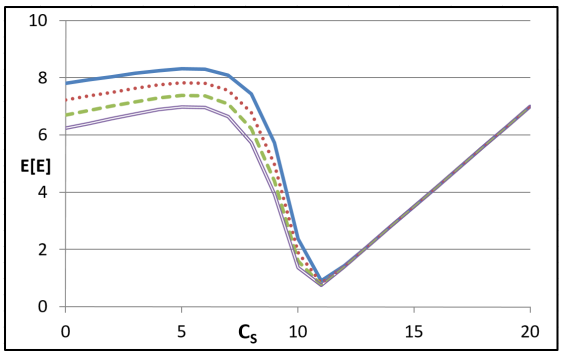
(e) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$



(f) $C = 50,\ \lambda = 25,\ \mu = 1,\ \gamma = 0.001$, with larger k



(g) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$



(h) $C = 20,\ \lambda = 10,\ \mu = 1,\ \gamma = 0.001$, with larger k

Figure 6.20: Expected energy consumption rate vs C_S for $\gamma = 0.001$

Chapter 7

Aysmptotic Performance

While past chapters (mainly Chapters 5 and 6) studied the model under general parameters (no constraints on C, λ, μ, and γ), it is often insightful to analyse stochastic models under an *asymptotic regime*, i.e. analysing the system's limiting behaviours. This typically includes allowing certain parameters to approach infinity, or the system to approach instability (from the stable side). In turn, certain asymptotic regimes run the risk of trivializing the model in question, and granting no additional insight. For example, an energy-aware system under a *heavy-load* asymptotic regime ($\rho \to 1$ from below), has its optimal policy degenerate to a "keep all servers on all the time" policy. This follows from the observation that the steady state probability that any server is idle approaches zero under such a regime.

There are however, asymptotic regimes which do grant insight. Consider an energy-aware system under a *fixed-load, many-server* asymptotic regime. Here the number of servers approaches infinity ($C \to \infty$) while the load remains fixed on some constant ($\rho = \lambda/(C\mu)$). It is not obvious what an optimal policy would behave like under this

regime for a given cost function, or if such a regime eases analysis. This is because there are an infinite number of servers which in practical terms are not needed, and therefore have the possibility of incurring infinite (unnecessary) energy costs. Moreover, due to the availability of these servers, the potential for the system to thrash cannot be ignored. For example, consider the case where a significantly large number of jobs accumulates in queue causing a large portion of these servers to be switched on. This then causes an over provisioning of processing power, which strongly pushes back on the accumulating jobs causing many servers to turn off which are usually busy. Having many servers turn off in the short run could cause a significantly large number of jobs to build up in the system and the cycle repeats. Therefore, this fixed-load, many-server regime offers non-trivial challenges, while potentially easing analysis.

Whereas in Chapter 6 the analysis produced results under a wide range of cost functions (identifying situations where $\mathbb{E}[R]$ and $\mathbb{E}[E]$ are both close to their minimum values), this chapter looks to extend this generalization to a wide range of policies. Before this is done however, it is important to note that while in Chapter 6 $\mathbb{E}[E]$ denoted the expected *excess* energy cost (due to convenience), for the remainder of this chapter $\mathbb{E}[E]$ will again denote the expected *total* energy cost (again, this is done due to convenience of future analysis). With that small consideration out of the way, two classes of policies are defined below.

Definition 7.1. ***Class A Policy:*** *A policy is said to be a Class A policy if the following conditions are met:*

1. *Server setups are invoked following a threshold scheme.*

2. *A server will never turn off if there is a job which it could be processing.*

Before the second class of policies is given, another definition must first be introduced. Because this work examines energy-aware systems as $C \to \infty$ it may be the case that while a policy gives a criterion to turn on some server s, the probability of this criterion being met approaches 0. Therefore to reason about these cases, and others, the following framework is introduced. Let $X_{\mathcal{E}}(s,t)$ be an indicator function such that

$$X_{\mathcal{E}}(s,t) = \begin{cases} 1, & \text{if server } s \text{ is in energy state } \mathcal{E} \text{ at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathcal{E} \in \{\textit{off, setup, idle, busy}\}$. Then it is said that s is an always $\mathcal{E}$ server if and only if as $t \to \infty$, $P(X_{\mathcal{E}}(s,t) = 1) \to 1$. As an example, if a server s has a criterion which turns it on and it is known that the server will always eventually turn off, but the probability that the turn on criterion is met approaches 0 as $t \to \infty$, s would be called an always off server, since as $t \to \infty$, $P(X_{\textit{off}}(s,t) = 1) \to 1$. With these notions in mind the second class of policies is defined as follows.

Definition 7.2. ***Class B Policy:*** *A policy is said to be a Class B policy if the following conditions are met:*

1. *It is a Class A policy.*
2. *There exists an $\alpha < 1$ such that the number of always idle servers is less than $(1-\rho)C^{\alpha}$.*
3. *For all n_1 and n_2, if a server s turns off when there are n_1 servers on and n_2 jobs in the system, then while there are at least n_1 servers on, s will not begin its setup until there are at least $n_2 + 1$ jobs in the system.*

The second condition for Class B policies states that the number of servers which are always idle cannot be on the same order as the total number of servers. The third condition protects against known suboptimal behaviour (and is necessary for the proof), see Theorem 5.3. That is, it is never the case that when a server switches off it immediately begins its setup process. It is worth noting that most policies studied in the literature are Class B policies, i.e. the policies of focus in [14, 15, 16, 20, 38, 42, 56, 58, 61, 66] are all Class B policies. The sets of Class A and Class B policies are denoted by Π_A and Π_B respectively, and furthermore, for a specific policy π, $\mathbb{E}[R^\pi]$ and $\mathbb{E}[E^\pi]$ denote the expected response time and expected energy costs under policy π, respectively.

Theorem 7.1. *All policies in Π_A are asymptotically optimal with regards to expected response time. In other words, given an energy-aware system, for any $\pi_a \in \Pi_A$, as $\lambda, C \to \infty$ and $\lambda/(\mu C)$ is fixed to be ρ, where $0 < \rho < 1$, $\mathbb{E}[R^{\pi_a}] \to 1/\mu$.*

While perhaps surprising at first, such a result becomes intuitive as one considers the details of the system behaviour. Informally, there is a significant proportion of jobs which are served immediately on arrival, and therefore a significant proportion of jobs have a response time equal to their service time. And while it is true that some jobs will have to wait to be served, whether it be for a server to complete a job or finish a setup, the number of these jobs turns out to be negligible under the asymptotic regime. It is worth noting that Theorem 7.1 would not necessarily hold for policies which turned servers off while there are waiting jobs that could be processed. On the other hand, belonging to Π_A is not necessary for minimizing the expected response time. With optimal policies now known for $\mathbb{E}[R]$, the focus shifts to the second cost metric, $\mathbb{E}[E]$. Recall that $\mathbb{E}[E^J]$ denotes the expected energy cost for a single job, and

furthermore, $\mathbb{E}[E^{J,\pi}]$ denotes the expected energy cost for a single job under policy π.

Theorem 7.2. *All policies in Π_B are asymptotically optimal with regards to expected energy cost. In other words, given an energy-aware system, for any $\pi_b \in \Pi_B$, as $\lambda, C \to \infty$ and $\lambda/\mu C$ is fixed to be ρ, where $0 < \rho < 1$, $\mathbb{E}[E^{\pi_b}]/\lambda \to \mathbb{E}[E^{J,\pi_b}] \to E_{Busy}/\mu$.*

Corollary 7.1. *All Class B policies are asymptotically optimal under any well-formed cost function.*

The optimality result for the expected energy cost is arguably more surprising than the result for the expected response time. One may have the intuition that some of these policies would regularly have an infinite number of servers in a specific energy state, such as setup, which would in turn incur an infinite amount more cost than some other policy. For example, one policy may regularly have an infinite number of servers in setup, while another may instead have an infinite number of servers idle. Considering these two policies, it may be fair to think one would incur infinitely more cost than the other, which makes the notion of these policies being equivalent under all cost functions difficult to wrap one's mind around. While this line of thinking is not contradicted by Theorem 7.2, it does say that focusing on details of servers which do spend a certain amount of time in setup or idling is a misleading way to think about these systems. In other words, under the asymptotic regime, although there are an infinite number of servers idle and/or in setup under many policies, the probability of a server being idle or in setup is zero. On the other hand, the probability of a server being off or busy is one. That is, the number of servers which are always busy and off is on the order of C; whereas the number of servers which are idle

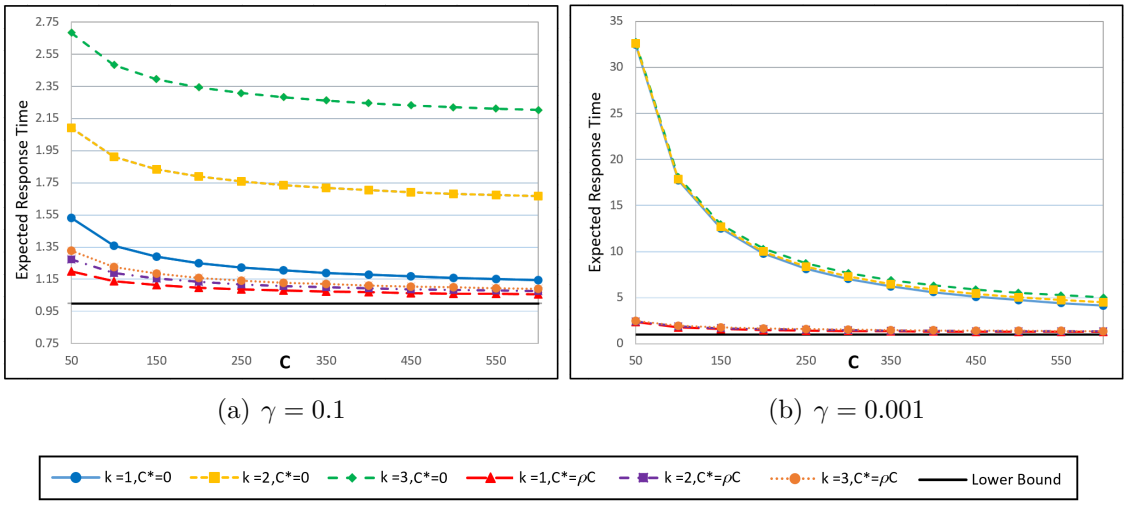
or in setup is on some order less than C, albeit still potentially infinite. One possible example would be the case where the number of servers idle is on the order of $\sqrt{C}$.

As will be seen in Section 7.1 and Appendix C, reasoning about the energy cost from the perspective of the servers, as was seen in (4.2), results in complexities which are washed away when the energy costs are viewed from the perspective of the jobs, i.e. $\mathbb{E}[E] = \lambda\mathbb{E}[E^J]$. Once these observations are made, these seemingly complex systems become simple to reason about.

The most significant implication of Theorems 7.1 and 7.2 is that under the asymptotic regime the trade-off between $\mathbb{E}[R]$ and $\mathbb{E}[E]$ is in fact not a trade-off at all. That is, not only are both cost metrics minimized across a large set of policies, but over all well-formed cost functions. This is a powerful result, since if a system is *close* to this asymptotic regime, then a manager can confidently employ a Class B policy knowing that it will be reasonably close to optimal. Of course this begs the question, what does it mean for a system to be *close* to the system of study? This work addresses this question by numerically inspecting energy-aware systems with a finite number of servers C, to see how quickly the cost metrics approach their bounds described in Theorems 7.1 and 7.2.

7.1 Numerical Experiments

All numerical experiments presented here are done for an energy-aware system employing a *staggered threshold* policy, with a specific instantiation of its decision variables k, and C_S. See Section 6.4 for details of the policy and method of analysis.



(a) $\gamma = 0.1$ (b) $\gamma = 0.001$

Figure 7.1: Expected response time vs C for $\lambda = C/2$, $\mu = 1$

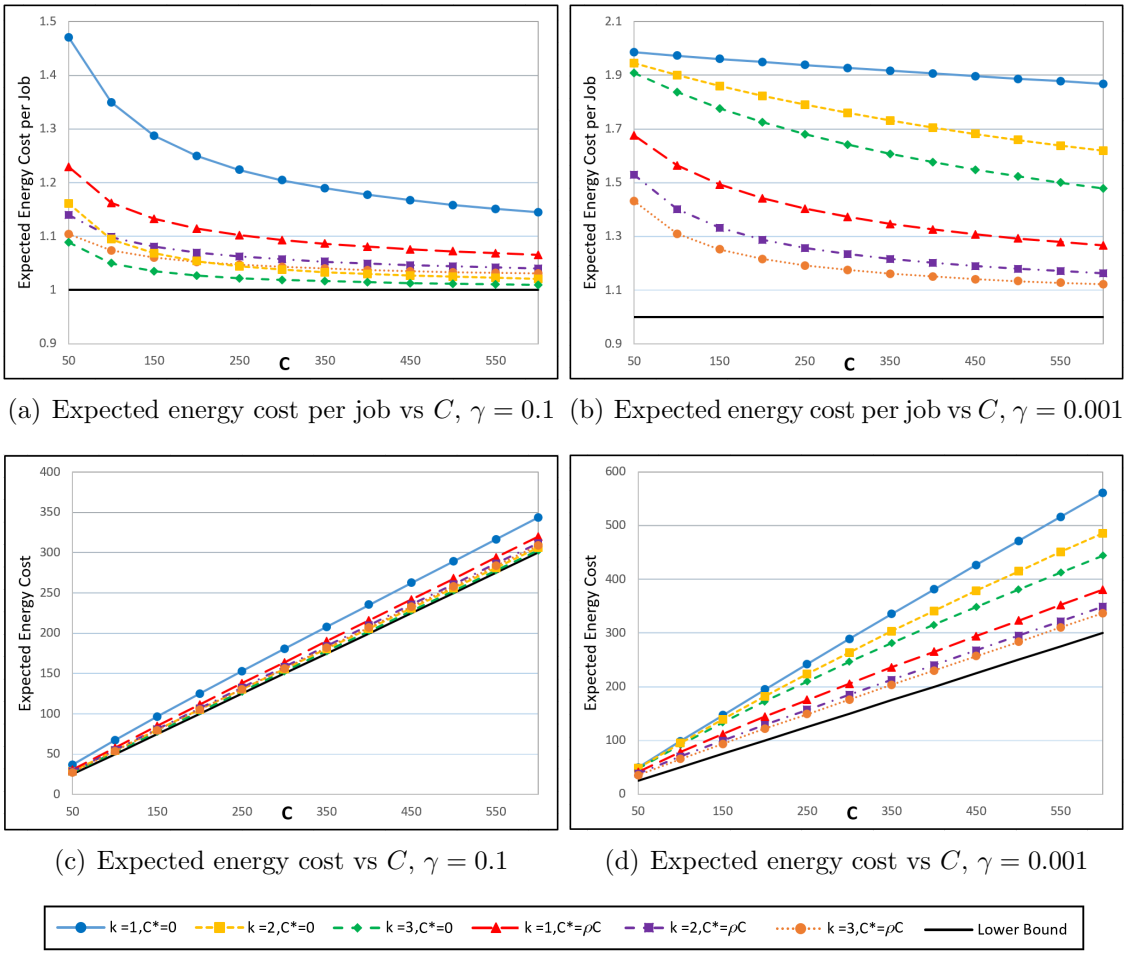
The purpose of these numerical experiments is firstly to ensure that exact analysis agrees with the results pertaining to the system under the asymptotic regime, and secondly to examine how quickly the system approaches the corresponding optimal behaviour as the parameters are appropriately scaled up.

Figure 7.1 shows the behaviour of $\mathbb{E}[R]$ as the system is scaled up. A preliminary observation is that for the curves where $C_S = 0$, the corresponding value of $\mathbb{E}[R]$ can be far from optimal even for large values of C. As an example, the curve where $k = 3$ and $C_S = 0$ in Figure 7.1-(a) is more than double that of the optimal value even for the largest values of C analysed. And perhaps even worse than that, the curves where $C_S = 0$ have extremely slow convergence rates. On the other hand, one may also note that when $C_S = \rho C$, $\mathbb{E}[R]$ becomes reasonably close to its optimal value relatively quickly. This effect is accentuated further in Figure 7.1-(b), where the setup times

are large. Here, all curves which share the same choice of C_S are visually grouped together, and moreover, when $C_S = 0$ the expected response time can be far from optimal even for larger values of C. But the curves which have a number of servers which are forced to be on, i.e. $C_S = \rho C$, gets much closer to the minimum value. In other words, the convergence rate is sensitive to the choice of C_S, while relatively insensitive to the threshold value k, especially when setup times are large. Forcing the system to have a number of servers always on, where the number always on equals ρC, causes the system to more quickly approach its minimum expected response time.

The appealing choice of forcing $\lambda/\mu = \rho C$ servers to always remain on is interesting, since as can be seen in Appendix C, the number of servers which are always busy approaches $\lambda/\mu = \rho C$ under the asymptotic regime. In other words, when the system parameters are finite, setting $C_S = \rho C$ forces the system to behave in a manner in which it is known to behave under the asymptotic regime. As such, it is intuitive that when the system is constrained to invoke certain asymptotic behaviour, i.e. $C_S = \rho C$, the corresponding values of $\mathbb{E}[R]$ are closer to values which would be seen under the asymptotic regime.

Shifting focus to the expected energy costs per job and Figures 7.2 (a) and (b), a similar trend regarding the choice of C_S is seen. That is, when C_S is forced to take on the value it practically approaches under the asymptotic regime, $\mathbb{E}[E^J]$ gets closer to its optimal value. Here plotting the expected energy cost on a per job basis allows one to clearly see how the system is becoming more efficient as the parameters are scaled up. This is in contrast to viewing the total expected energy cost as seen in



(a) Expected energy cost per job vs C, $\gamma = 0.1$ (b) Expected energy cost per job vs C, $\gamma = 0.001$

(c) Expected energy cost vs C, $\gamma = 0.1$ (d) Expected energy cost vs C, $\gamma = 0.001$

Figure 7.2: Expected energy cost and expected energy cost per job vs C for $\lambda = C/2$, $\mu = 1$

Figure 7.2 (c) and (d). Here one in fact sees that the curves are diverging from their lower bound. At first this may seem to be a direct contradiction of Theorem 7.2 due to the observed divergence, but this is not the case. While it is true that the difference between the expected energy costs and the optimal value is getting larger, it is important to remember that the lower bound itself is also growing with the system

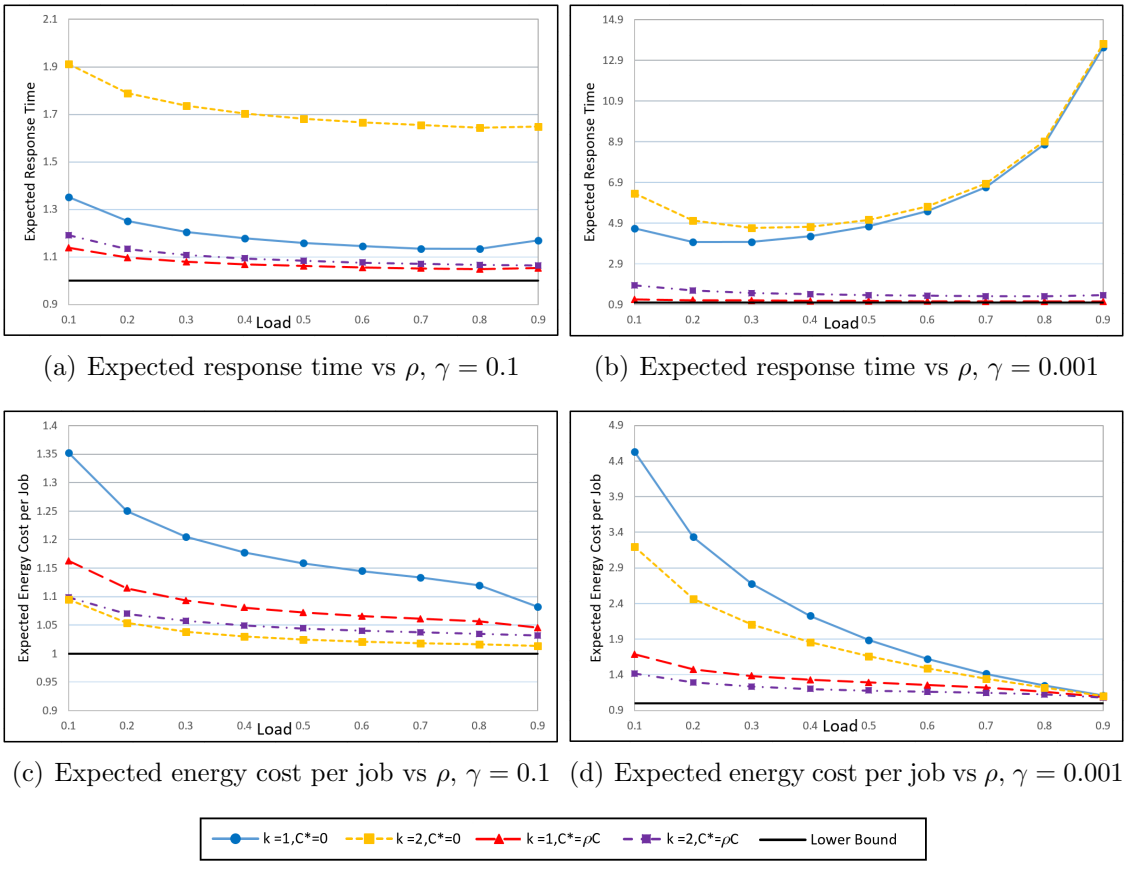
parameters, specifically λ. In other words, as $C \to \infty$ the difference between the energy costs and lower bound is growing to infinity, but is growing sub-linearly. At the same time, the lower bound is growing to infinity linearly. Therefore, the difference between the expected energy cost and the lower bound, while infinite, becomes negligible when compared to the total cost. One of the key insights that looking at the per job expectations of the energy cost grants under this scaling, is that a divergence from the lower bound of the total expected energy cost does not imply suboptimality, specifically under the asymptotic regime.

Another aspect of these systems which warrants attention, is how sensitive the convergence rate is to the load. This is seen in Figure 7.3. Examining the load's effect on the expected response time, one can observe that when the setup times are relatively short, the convergence rate is relatively insensitive to choice of load. Furthermore, the most sensitive parts of the curves are when the load is light or heavy, especially in Figure 7.3 (b) where the setup times are longer. This makes some intuitive sense, since when the loads are light or heavy the system is more likely to exhibit behaviour that is not described by the asymptotic regime. When the load is light, the system has a significant chance to be empty, and in turn has a significant chance to have the minimum number of servers on. When jobs arrive it begins to overcompensate with more setups than are needed and the servers begin to thrash. On the other hand when the system load is high there is a significant chance that there will be more than C jobs in the system. So even if all servers were on, jobs would still have to wait. Having many servers regularly thrashing, or having more jobs in the system than servers are two characteristics which are never exhibited by a system under the

asymptotic regime. Therefore, it is intuitive that a system under light or heavy loads would be slower to exhibit asymptotic behaviour than a system with a medium load.

With this sensitivity in mind, one can still clearly note that the previous observation regarding having ρC servers always on induces the asymptotic behaviour to occur sooner. The only curve not to agree with this notion is the case where $k = 2$ and $C_S = 0$ in Figure 7.3 (c). In this case the system is approaching the minimum value $\mathbb{E}[E^J]$ i.e. E_{Busy}/μ, slightly quicker than the curves where $C_S = \rho C$. This is a product of the servers thrashing, causing most jobs to see the system when many other jobs are present and therefore little energy is wasted, but is only achieved at the cost of a significant increase in $\mathbb{E}[R]$, and therefore this configuration would not be suggested.

Overall the numerical experiments offer favourable results. That is the optimality of policies belonging to the set Π_B can be seen in systems with a finite number of servers relatively quickly. Moreover, modern data-centres have a number of servers on the order of tens of thousands (likely on the order of hundreds of thousands for tech giants such as Google, Amazon, etc.). The experiments ran here were only for values on the order of hundreds (due to computational constraints). Therefore, one could apply this observation to a practical setting with a reasonable degree of confidence. Further still, this static provisioning around ρC to induce asymptotic behaviour strongly agrees with the key observations from Chapter 6. That is, in Chapter 6 for it was seen that for systems with a finite number of servers, a static provisioning around $\rho C + \sqrt{\rho C}$ was near optimal for both metrics. But of course as C gets large $\rho C + \sqrt{\rho C}$ and ρC become approximately equal relatively quickly.



(a) Expected response time vs ρ, $\gamma = 0.1$ (b) Expected response time vs ρ, $\gamma = 0.001$

(c) Expected energy cost per job vs ρ, $\gamma = 0.1$ (d) Expected energy cost per job vs ρ, $\gamma = 0.001$

Figure 7.3: Expected response time and expected energy cost per job vs ρ for $C = 500$, $\mu = 1$

Chapter 8

Conclusion

Energy consumption of modern day data-centres is immense, and growing. One of the primary concerns when dealing with these systems is when should one turn an idle server off (to attempt to save operating costs) and when should one then turn that server back on (to improve system performance). Due to uncertainties such as when will the next job arrive, how long will it take to process, how long will a server take to turn on, etc., these aforementioned questions are far from trivial to answer.

This work presented an established model and identified several difficulties and potential shortcomings regarding its analysis. That is, due to the range of system parameters and choice of cost function, making claims on policies can be problematic, and making strong claims such as: a policy π is optimal under all well-formed cost functions seems impossible. To address these issues, this thesis took a three-pronged approach.

Firstly, using Markov decision processes (MDPs) the structure of the optimal policy was studied. Providing formal proofs of these structural results allowed for an exclusion of policies which were now immediately known to always be suboptimal. Secondly, from this now smaller set of policies, three further families of policies were defined and analysed, i.e. Bulk Setup, Dual Threshold, and Staggered Threshold. From an exact analysis, a suite of numerical experiments were conducted granting several key insights. Of these insights, arguably the two most important are 1) that scenarios exist where $\mathbb{E}[R]$ and $\mathbb{E}[E]$ are simultaneously close to their corresponding minimums, and 2) that while policies may seem to be quite different by definition, their overall behaviour can be quite similar (see Bulk Setup and Staggered Threshold). Lastly, to explore this observation of policies behaving similarly, the model is analysed under a fixed-load, many-server asymptotic regime, where $C \to \infty$, $\lambda \to \infty$, while $\lambda/(C\mu)$ is fixed. Here, it is formally proven that a large set of policies, including the vast majority which are studied in the literature, become equivalent and optimal under this regime. Numerical studies verified these results, and also granted insights into how to induce these optimality results sooner.

8.1 Future Work

As is often the case with research, providing results and solving problems gives rise to even more questions; this thesis is no exception. There are many fruitful directions future research could take. Perhaps the most obvious is the relaxation of model assumptions. For example, while interarrival times being exponentially distributed is a justifiable assumption (this follows from an analytic result showing as the number of mutiplexed renewal processes increases, the aggregate process weakly converges to a

Poisson process [28]). The imposed assumption on the service and setup times also being exponentially distributed is more difficult to justify. Service times have been studied to often times be heavy tailed, and typical experience can contradict exponentially distributed turn on times.

Of course, these assumptions were made for analytic tractability, which begs the question, what can one do assuming general distributions rather than exponential? At the risk of sounding pessimistic, the hope of performing an exact analysis on the model is minimal (if not completely gone). This intuition follows from the fact that much simpler queueing models with general distributions still remain to be solved, e.g. the $M/G/1$ queue. However, some things can still be done. Of course one could always fall back on simulation to empirically verify if the observations offered in this thesis still hold with the distributional assumptions relaxed. But other options exist as well, specifically, if the setup times are allowed to follow an arbitrary distribution, while exact analysis may still be challenging, it is the author's opinion that the asymptotic performance results could be extended under a further restriction to the set of policies. That is, if a subset of Class B policies were to be defined, say Class C policies, where each arriving job could cause at most a finite number of servers to enter setup, then there is a strong hope that Theorem 7.2 would still hold; of course, the devil is in the details.

There is also intuition that some of the structural properties should still hold as well. Specifically, there is no obvious reason that general distributions for both the service and setup times would now cause one to want to turn a server off while there

are jobs in the system. The immediate challenge is that the proof of this structural property is built around renewal arguments which exploit the memoryless property of the exponential distribution. However, one may be able to cleverly choose certain system states of which a cycle still qualifies as a renewal process. For example, again assume that service times are exponentially distributed while setup times are arbitrarily distributed. Here, any state which does not have servers in setup is memoryless. Furthermore, because server setups are interruptible, the only time one would want to consider turning a server off is in a state where no servers are currently in setup. Therefore, it would stand that a similar proof may exist for a model with generally distributed setup times.

While relaxing the assumptions on the underlying distributions may be an obvious next step, there is another assumption which is somewhat more subtle. When defining the expected energy cost, $\mathbb{E}[E]$, it is assumed that the cost is linear in the number of servers in certain states. As an example, if a two-server system has one server off and one server in setup, and another two-server system has both servers in setup, the current energy cost of the latter would be twice that of the former. This at first seems like a fair assumption. However, it may not be. It is true that the energy used by these systems is likely close to linear in the number of servers in setup, but the energy cost could be highly non-linear. That is, energy providers charge more for clients who are using more energy at a given time. This has many interesting implications to how management of these systems could change. As an example, putting multiple servers into setup (such as in the bulk setup policy), may be incredibly unappealing compared to putting servers into setup one at a time. From the analytic side of things, again,

relaxing such an assumption would be challenging. To make the model as general as possible one could define $\mathbb{E}[E]$ abstractly, and assume that it is monotonically increasing in the number of busy, idle, and setup servers. Exact analysis becomes difficult if this is done, but some structural properties may still hold. Moreover, if one imposed a ceiling on the energy cost (for instance, the maximum rate which a provider would charge) the asymptotic performance may also be extendable. One instant benefit of abstracting the definition of $\mathbb{E}[E]$ is that the model can exactly capture the behaviour of speed scaling models (discussed in Chapter 3) by appropriately instantiating the definition of $\mathbb{E}[E]$ and letting $\gamma \to \infty$.

Moving past relaxing assumptions and looking at further analysis which could be performed on the model as is, is also interesting. One issue that exists in these systems, but is rarely ever addressed, is that of fairness. As an example, consider an energy-aware system with a single server which turns the server off the moment it idles and begins turning it on when ten jobs are waiting in the queue. It is completely reasonable for a policy such as this to be optimal for some set of system parameters and cost function. Furthermore, such a policy could also have reasonable performance, i.e. $\mathbb{E}[R]$ may be close to $1/\mu$. However, consider the differing experiences of a job which arrives to an empty system, and one which arrives to a system where the server is on. The job arriving to an empty system not only has to wait for the setup process of the server, but also has to wait for nine more jobs to arrive to the system before that setup process even begins. From the assumption that $\lambda < \mu$, this implies that job will expect to wait at least $9/\mu$ (perhaps much longer), whereas *lucky* jobs like the one arriving to a system where the server is already on, could wait much less time. It is

true that in all stochastic models some jobs will have a better experience than others, due to uncertainty, but here it seems these effects may be exaggerated. As such, this issue of vastly different response times (which may also be seen quite frequently), may deserve some attention.

One popular criticism of the model (and ones similar to it) is the arrival rate is constant, whereas in practice data-centres historically experience heavier and lighter loads throughout the day. This is a well known open question in the community. However, it is the author's opinion that if the change in arrival rate is on a slow enough time scale, say hours, while other system parameters/events are on the order of seconds or less (interarrival time, service times, setup times), then the observations provided in this work are still applicable, but may need to be updated on the aforementioned macro time scale.

It is clear that there is no shortage of potential directions this research can take, and hopefully the material presented in this section has highlighted some of those directions. Having said that, the results provided in this work are substantial, novel, and insightful. Using higher level theoretical approaches this work developed, derived, and presented conclusions that are practical to timely real world issues.

Bibliography

[1] Source code. `http://www.cas.mcmaster.ca/~macciov/publications.html`. Accessed: 2016-03-01.

[2] I. Ahmed and S. Ranka. *Handbook of Energy-Aware and Green Computing - Two Volume Set*. CRC Press, 2016.

[3] A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

[4] J. R. Artalejo. A unified cost function for M/G/1 queueing systems with removable server. *Trabajos de Investigacion Operativa*, 7(1):95–104, 1992.

[5] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[6] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012.

[7] M. Callau-Zori, L. Arantes, J. Sopena, and P. Sens. MERCi-MIsS: Should I turn

off my servers? In *Distributed Applications and Interoperable Systems*, pages 16–29, 2015.

[8] M. Caramia and S. Giordani. Resource allocation in grid computing: An economic model. *WSEAS Transactions on Computer Research*, 3(1):19–27, 2008.

[9] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. *SIGMETRICS Performance Evaluation Review*, 33(1):303–314, 2005.

[10] M. Elahi and C. Williamson. Autoscaling effects in speed scaling systems. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 307–312, Sept 2016.

[11] M. Elahi, C. Williamson, and P. Woelfel. Decoupled speed scaling: Analysis and evaluation. *Performance Evaluation*, 73:3 – 17, 2014. Special Issue on the 9th International Conference on Quantitative Evaluation of Systems.

[12] EPA. Report to congress on server and data center energy efficiency. Technical report, U.S Environmental Protection Agency, 2007.

[13] S. W. Fuhrmann and R. B. Cooper. Stochastic decompositions in the M/G/1 queue with generalized vacations. *Operations Research*, 33:1117–1129, 1985.

[14] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf. Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. In *ACM SIGMETRICS*, 2013.

[15] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch. Optimality analysis of energy-performance trade-off for server farm management. *Performance Evaluation*, 67(11):1155–1171, 2010.

[16] A. Gandhi and M. Harchol-Balter. M/M/k with exponential setup. Technical report, Carnegie Mellon University, 2010.

[17] A. Gandhi and M. Harchol-Balter. How data center size impacts the effectiveness of dynamic power management. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011.

[18] A. Gandhi and M. Harchol-Balter. M/G/k with staggered setup. *Operations Research Letters*, 41(4):317 – 320, 2013.

[19] A. Gandhi, M. Harchol-Balter, and I. Adan. Decomposition results for an M/M/K with staggered setup. *SIGMETRICS Performance Evaluation Review*, 38(2):48–50, Oct. 2010.

[20] A. Gandhi, M. Harchol-Balter, and I. Adan. Server farms with setup costs. *Performance Evaluation*, 67(11):1123–1138, 2010.

[21] A. Gandhi, M. Harchol-Balter, and M. A. Kozuch. Are sleep states effective in data centers? In *Proceedings of the 2012 International Green Computing Conference (IGCC)*, IGCC '12, pages 1–10, Washington, DC, USA, 2012. IEEE Computer Society.

[22] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch. Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems*, 30(4):14:1–14:26, Nov. 2012.

[23] M. E. Gebrehiwot, S. Aalto, and P. Lassila. Optimal sleep-state control of energy-aware M/G/1 queues. In *8th International Conference on Performance Evaluation Methodologies and Tools*, 2014.

[24] M. E. Gebrehiwot, S. Aalto, and P. Lassila. Energy-aware control of server farms. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 748–753, May 2016.

[25] M. E. Gebrehiwot, S. Aalto, and P. Lassila. Energy-aware server with SRPT scheduling analysis and optimization. In *Lecture notes in computer science*, pages 107–122, 2016.

[26] M. E. Gebrehiwot, S. Aalto, and P. Lassila. Energy-performance trade-off for processor sharing queues with setup delay. *Operations Research Letters*, 44(1):101 – 106, 2016.

[27] D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. Wiley-Interscience, Third edition, 1998.

[28] M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[29] P. G. Harrison, N. M. Patel, and W. J. Knottenbelt. Energy–performance trade-offs via the EP queue. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 1(2):6:1–6:31, June 2016.

[30] M. Hassan and M. Atiquzzaman. A delayed vacation model of an M/G/1 queue

with setup time and its application to SVCC-based ATM networks. *IEICE Transactions on Communications*, pages 317–323, 1997.

[31] J. Hu and T. Phung-Duc. Power consumption analysis for data centers with independent setup times and threshold controls. In *AIP*, 2015.

[32] E. Hyytiä and R. Righter. Fairness through linearly increasing holding costs in systems of parallel servers with setup delays. In *2015 27th International Teletraffic Congress*, pages 143–151, Sept 2015.

[33] E. Hyytiä, R. Righter, and S. Aalto. Energy-aware job assignment in server farms with setup delays under LCFS and PS. In *2014 26th International Teletraffic Congress (ITC)*, pages 1–9, Sept 2014.

[34] E. Hyytiä, R. Righter, and S. Aalto. Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. *Performance Evaluation*, 75:17 – 35, 2014.

[35] D. L. Iglehart. Limiting diffusion approximations for the many server queue and the repairman problem. *Journal of Applied Probability*, 2(2):429–441, 1965.

[36] L. Kleinrock. *Queueing Systems*, volume One. Wiley-Interscience, 1975.

[37] J. Koomey. Growth in data center electricity use 2005 to 2010. A report by Analytical Press, completed at the request of The New York Times, `http://www.analyticspress.com/datacenters.html`, 2011.

[38] P. J. Kuehn and M. E. Mashaly. Automatic energy efficiency management of data center resources by load-dependent server activation and sleep modes. *Ad Hoc Networks*, 25(2):497–504, 2015.

[39] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew. Greening geographical load balancing. In *the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS, pages 233–244, 2011.

[40] V. J. Maccio. On optimal policies for energy-aware servers. Master's thesis, McMaster University, 2013.

[41] V. J. Maccio and D. G. Down. Asymptotic performance of energy-aware multiserver queueing systems with setup times. Technical Report CAS-16-05-DD, Department of Computing and Software, McMaster University.

[42] V. J. Maccio and D. G. Down. Exact analysis of energy-aware multiserver queueing systems with setup times. Technical Report CAS-16-01-DD, Department of Computing and Software, McMaster University.

[43] V. J. Maccio and D. G. Down. On optimal control for energy-aware queueing systems. Technical Report CAS-15-05-DD, Department of Computing and Software, McMaster University.

[44] V. J. Maccio and D. G. Down. On optimal policies for energy-aware servers. In *the IEEE 21st International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 31–39, 2013.

[45] V. J. Maccio and D. G. Down. On optimal control for energy-aware queueing systems. In *27th International Teletraffic Congress (ITC 27)*, pages 98–106, 2015.

[46] V. J. Maccio and D. G. Down. On optimal policies for energy-aware servers. *Performance Evaluation*, 90:36 – 52, 2015.

[47] V. J. Maccio and D. G. Down. Exact analysis of energy-aware multiserver queueing systems with setup times. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 11–20, Sept 2016.

[48] M. J. Magazine. On optimal control of multi-channel service systems. *Naval Research Logistics Quarterly*, 18(2):429–441, 1971.

[49] M. Mazzucco and D. Dyachuk. Optimizing cloud providers revenues via energy efficient server allocation. *Sustainable Computing: Informatics and Systems*, 2(1):1–12, 2012.

[50] I. Mitrani. Managing performance and power consumption in a server farm. *Annals of Operations Research*, 202(1):121–134, 2013.

[51] T. H. Nguyen, M. Forshaw, and N. Thomas. Operating policies for energy efficient dynamic server allocation. *Electronic Notes in Theoretical Computer Science*, 318:159–177, 2015.

[52] D. Paul, W. D. Zhong, and S. K. Bose. Energy efficient scheduling in data centers. In *2015 IEEE International Conference on Communications (ICC)*, pages 5948–5953, June 2015.

[53] Y. Peng, D. K. Kang, F. Al-Hazemi, and C. H. Youn. Energy and QoS aware resource allocation for heterogeneous sustainable cloud datacenters. *Optical Switching and Networking*, Preprint, 2016.

[54] T. Phung-Duc. Multiserver queues with finite capacity and setup time. In *Analytical and Stochastic Modelling Techniques and Applications: 22nd International Conference*, pages 173–187, 2015.

[55] T. Phung-Duc. Single-server systems with power-saving modes. In *Analytical and Stochastic Modelling Techniques and Applications: 22nd International Conference*, pages 158–172, 2015.

[56] T. Phung-Duc. Exact solutions for M/M/c/setup queues. *Telecommunication Systems*, 64(2):309–324, 2017.

[57] T. Phung-Duc. Single server retrial queues with setup time. *Journal of Industrial and Management Optimization*, 13(3):1329–1345, 2017.

[58] T. Phung-Duc and K. Kawanishi. Energy-aware data centers with s-staggered setup and abandonmen. In *Analytical and Stochastic Modelling Techniques and Applications: 23rd International Conference*, pages 269–283, 2016.

[59] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[60] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. *ACM SIGCOMM Computer Communication Review*, 39(4):123–134, 2009.

[61] Y. Ren, T. Phung-Duc, J. C. Chen, and Z. W. Yu. Dynamic auto scaling algorithm (DASA) for 5G mobile networks. In *Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[62] D. R. Smith and W. Whitt. Resource sharing for efficiency in traffic systems. *Bell System Technical Journal*, 60(1):39–55, 1981.

[63] N. Tian and Z. G. Zhang. A two threshold vacation policy in multiserver queueing systems. *European Journal of Operational Research*, 168(1):153–163, 2006.

[64] N. Tian and Z. G. Zhang. *Vacation Queueing Models - Theory and Applications*. Springer Science, 2006.

[65] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation*, 69(12):601–622, Dec. 2012.

[66] X. Xu and N. Tian. The M/M/c queue with (e, d) setup time. *Journal of Systems Science and Complexity*, 21(3):446–455, 2008.

[67] M. Yajima and T. Phung-Duc. Batch arrival single-server queue with variable service speed and setup time. *Queueing Systems*, pages 1–20, 2017.

[68] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li. Stackelberg game approach for energy-aware resource allocation in data centers. *IEEE Transactions on Parallel and Distributed Systems*, 27(12):3646–3658, 2016.

Appendices

A Proofs for Theorems of Chapter 5

Before this work presents the proofs in Chapter 5, some notation must first be introduced. Using the MDP given in (5.3) one can define the traditional optimality equations as follows

$$\begin{aligned} w^* + u(i,j,m) = \min_{a \in A_s} \{ & w(i,j,m,a) + \lambda u(i + \min(0,a), j+1, m + \max(0,a)) \\ & + (m + \max(0,a))\gamma u(i + \min(0,a) + 1, j, m + \max(0,a) - 1) \\ & + \min(j, (i + \min(0, a+m)))\mu u(i + \min(0,a), j-1, m + \max(0,a)) \\ & + [1 - \lambda - (m + \min(0,a))\gamma - (i + \max(0,a))\mu] \\ & \cdot u(i + \max(0,a), j, m + \min(0,a))\} \end{aligned} \tag{1}$$

where w^* is the optimal average cost. Similarly, the corresponding optimality equation can be defined for the MDP given in (5.4).

$$\begin{aligned} w^* + u(i,j) = \min_{a \in A_s} \{ & w((i,j),a) + \lambda u(i + \min(0,a), j+1) + (\max(0,a))\gamma u(i+1, j) \\ & + \min(j, (i + \min(0,a)))\mu u(i + \min(0,a), j-1) \end{aligned}$$

$$+ (1 - \lambda - \min(0, a)\gamma - (i + \max(0, a))\mu)u(i + \max(0, a), j)\}, \tag{2}$$

where w^* again, is the optimal average cost. These equations are important specifically to the proof of Theorems 5.5 and 5.6. With these definitions aside, the proofs of the theorems presented in Chapter 5 are given.

A.1 Proof of Theorem 5.1

For simplicity of navigation Theorem 5.1 is restated.

Theorem 5.1. *For all energy-aware systems, the optimal policy is a* pure policy. *That is at every decision epoch, the optimal policy will never turn a non-zero number of servers off and put a non-zero number of servers into setup.*

Proof. This theorem is shown using a sample path argument, exploiting the fact that a server consumes energy at a lesser rate while idle, than in setup. Consider a system in state (i, j, m) which puts n_1 servers into setup, and turns n_2 servers off, where $0 < n_1 \leq (C - i - m)$ and $0 < n_2 \leq i$. This would now put the system in state $(i - n_2, j, m + n_1)$ until the next decision epoch. If this behaviour can be shown to always be suboptimal, then the theorem is proven. The following two cases show this is indeed true.

In Case 1, suppose that $n_1 \geq n_2$. The system could have used a pure setup strategy by turning on $n_1 - n_2$ servers, putting the system in state $(i, j, m + (n_1 - n_2))$. It is easy to see that $u(i, j, m + (n_1 - n_2)) < u(i - n_2, j, m + n_1)$. This is due to the fact that the mixed system (the system which simultaneously switches servers on and off)

has n_1 more servers in setup than the system that only performs server setups as its current action. Although the pure setup strategy has n_2 more servers on than the mixed strategy, even in the worst case, $(j \leq i + n_2)$, these servers will still incur a lesser cost since the energy cost of idling is less than the energy cost of being in setup. It is also noted that in the best case $(j \geq i + n_2)$, the extra n_2 servers do not incur a cost at all since the energy required to process jobs will be used eventually. Once the next event occurs, the pure setup strategy can be changed back to a mixed strategy to synchronize the two systems. As a result, the pure strategy (only server setups) incurs lesser cost than the mixed strategy.

In Case 2, suppose that $n_1 < n_2$. As in Case 1, the system could have used a pure strategy but instead of putting servers into setup, it could just turn $(n_2 - n_1)$ servers off. This would put the system in state $(i - (n_2 - n_1), j, m)$. Similar to Case 1, it can be seen that $u(i - (n_2 - n_1), j, m) < u(i - n_2, j, m + n_1)$. The system employing the mixed strategy has n_1 extra servers in setup, while the system employing the pure strategy has an extra n_1 servers on. Again, even in the worst case of all of the additional i servers being idle, the pure strategy is still preferred since idle servers incur energy costs at a lower rate than those in setup. As for the previous case, the two systems can then be synchronized at the next decision epoch, leading to a strictly lesser cost for the pure strategy. Furthermore, the same arguments can be applied to an interruptible energy-aware system which has a two dimensional state space.

□

A.2 Proof of Theorem 5.2

For simplicity of navigation Theorem 5.2 is restated.

Theorem 5.2. *For all energy-aware systems, for all well-formed cost functions, the optimal policy is a threshold policy.*

Proof. To prove that the optimal policy is a threshold policy three properties must be shown. Firstly, optimal actions are always made the moment an event occurs. Secondly, any benefit of turning off the ith server in state $(i, j+1, m)$ is also gained by turning off the ith server in state (i, j, m). Lastly, any benefit gained by turning on the $(i+m+1)$th server in state (i, j, m) is also gained by turning on the $(i+m+1)$th server in state $(i, j+1, m)$.

The first property is an immediate consequence of all underlying distributions being exponential. As such, the proof moves to the second property (the threshold nature of the turn offs). This is shown via contraposition. That is, if it is optimal to keep the ith server on in state (i, j), then it is also optimal to keep the ith server on in state $(i, j+1)$, if and only if the second property holds.

Therefore, it is shown that any benefit which keeping the ith server on offers in state (i, j, m) will also be offered in state $(i, j+1, m)$. In fact, keeping the ith server on can be even more beneficial in state $(i, j+1, m)$. Firstly, if the ith server could process a job in state (i, j, m), then it can clearly process a job in state $(i, j+1, m)$ as there are more jobs in the system. Secondly, if the server were to idle in anticipation of a job arriving in state (i, j, m), then it could also idle in state $(i, j+1, m)$, with an even higher probability that it will be utilized in the future since there are more jobs

present. Lastly, there exists the case of $j = i - 1$, where in state (i, j, m) the server would idle, while in state $(i, j + 1, m)$ it would be able to process jobs. If however, for whatever reason in state $(i, j + 1, m)$ it would instead be better for the system for the ith server to idle rather than process the job (to mimic its behaviour in state (i, j, m)) this would also be an option. Therefore, any benefit (lesser cost incurred) which keeping on the ith server on offers in state (i, j, m) is also present in state $(i, j + 1, m)$.

To show that the turn on criteria also follows a threshold policy is slightly more complex, since unlike turn offs, setups do not happen instantaneously. As such a sample path argument is used to show that if it is optimal to turn on a server in state (i, j, m), then it is also optimal to turn said server on in state $(i, j + 1, m)$.

Consider two energy-aware systems, $S_1 = S_2 = (C, \lambda, \mu, \gamma)$, where S_1 is in state (i, j, m) and S_2 is in state $(i, j + 1, m)$. Furthermore, with the exception of the extra job in S_2, assume all future and current job sizes, setup times, and job arrival times to be the same in both systems. Assume S_1 employs an optimal policy and in state (i, j, m) it begins to turn on the $(i + m + 1)$th server. S_2 uses a policy which behaves as follows:

- S_2 dynamically *marks* any job that begins to or is currently being processed in S_1, as well as the extra job which was initially present
- All servers in S_2 will only process a job if it is marked, and will immediately begin to process any marked job which is waiting (a server will never turn off if there is a marked job to be processed)

- Each server in S_2 (denoted by $s_{2,i}$), dynamically inspects the corresponding server in S_1 (denoted by $s_{1,i}$), and behaves according to the following conditions:
  - If $s_{2,i}$ is off and $s_{1,i}$ enters setup, then $s_{2,i}$ immediately enters setup.
  - If there are no marked jobs waiting to be processed in S_2, $s_{2,i}$ is idle, and $s_{1,i}$ is off, then $s_{2,i}$ turns off.
  - If there are no marked jobs waiting to be processed in S_2, $s_{2,i}$ is idle, and $s_{1,i}$ is in setup, then $s_{2,i}$ remains idle.

Firstly, from this policy it is clear that there will never be more jobs in S_1 than in S_2, as S_2 is constrained to only process jobs which have been or are being processed by S_1. Secondly, because S_1 is employing an optimal policy, when a server turns off in S_2 and no new jobs have arrived since the corresponding server turned off in S_1, S_2 is performing an optimal action (the same number of jobs are present in S_2 as there were in S_1 when the server was turned off). On the other hand, if more jobs have arrived since the corresponding server in S_1 turned off, knowing this future information, it may have been optimal to keep the corresponding server in S_1 on. Therefore, it would now also be optimal to keep the server on in S_2. However, since S_1 is employing an optimal policy the total cost incurred is less than the total cost saved by turning the servers off. Moreover, it is never optimal to turn off a server in S_2 but keep the server on in S_1 due to the threshold nature of turn offs. Thirdly, while S_1 and S_2 are unsynchronized, the cost saved by turning on the $(i+m+1)$th server in S_2 is greater than or equal to the cost saved by turning on the $(i+m+1)$th server in S_1. This is due to the fact that in S_2 the $(i+m+1)$th server will process at least the amount of jobs it does in S_1 (along with the potential to also process the extra marked job), and

not incur any extra energy cost. The energy costs are less than or equal to those of S_1 since as any server in S_2 that idles in S_1 also idles or is in setup (setup incurs energy costs at a higher rate than idling). Lastly, S_1 and S_2 will synchronize once there are no longer any marked jobs in S_2 and all costs for the two systems moving forward will be equal (optimal). Therefore the overall cost to turn on the $(i+m+1)$th server in state $(i, j+1, m)$ is less than keeping it off, or in other words in state $(i, j+1, m)$ it is optimal to turn on the $(i+m+1)$th server. Furthermore, the same arguments can be used for a interruptible energy-aware system with a two dimensional state space. □

A.3 Proof of Theorem 5.3

For simplicity of navigation Theorem 5.3 is restated.

Theorem 5.3. *For all non-interruptible energy-aware systems, if while in the state $(i-1, j, m)$ it is optimal to begin turning a server on, then in state (i, j, m) it is sub-optimal to turn a server off. Furthermore, for all interruptible energy-aware systems if while in the state $(i-1, j)$ it is optimal to begin turning a server on, then in state (i, j) it is suboptimal to turn a server off.*

Proof. This proof is done via contradiction. Consider an energy-aware system where it is assumed 1) that in state $(i+1, j, m+a-1)$ it is optimal to turn a server off and 2) that in state (i, j, m) the optimal action is to turn a servers on, where $0 < a \leq C - i - m$. If it is shown that these two assumptions are inconsistent, then the theorem must hold. The second assumption implies that in state $(i, j, m+a-1)$ it is optimal to begin turning exactly one server on. Now consider the same system in state $(i+1, j, m+a-1)$. From the first assumption, this would immediately bring the system to state $(i, j, m+a-1)$, and since in this state the optimal action is

to turn a server on (as previously seen), a server would immediately be switched on bringing the system to state $(i, j, m + a)$. However, during that decision epoch the optimal policy was not a pure policy (servers were switched off and put into setup at the same time), which violates Theorem 5.1. Therefore, the optimal action in state $(i, j, m + a - 1)$ must be to do nothing, which implies that in state (i, j, m) turning a servers on is suboptimal, which violates the assumption. Furthermore, the same arguments can be applied to a two state interruptible energy-aware server. □

A.4 Proof of Theorem 5.4

For simplicity of navigation Theorem 5.4 is restated.

Theorem 5.4. *For all* $0 \leq i < (C-1)$ *and* $0 \leq j < (C-i-1)$, $k^{+}_{m+1,i} \leq k^{+}_{m,i+1}$, *and* $0 < i < C$ *and* $0 \leq j \leq (C-i)$, $k^{-}_{i,m+1} \leq k^{-}_{i+1,m}$ *and* $k^{-}_{i,m} \leq k^{-}_{i,m+1}$.

This proof shows each inequality, i.e. $k^{+}_{m,i} \leq k^{+}_{m+1,i-1}$, $k^{-}_{i,m-1} \leq k^{-}_{i,m}$ and $k^{-}_{i+1,m-1} \leq k^{-}_{i,m}$ for all valid m and i, by three individual lemmas. Before this is done however, a lemma regarding optimal turn on decision timing is firstly shown.

Lemma A.1. *It is optimal to begin turning on a server only at job arrival times.*

Proof. Due to the Markovian nature of the system it is known that decisions are only made the moment an event occurs. Events which can occur in the system are arrivals, departures, a server turning on or a server turning off. Choosing to turn a server on when another is turned off violates Theorem 5.1. Choosing to turn a server on at a departure violates Theorem 5.2. Therefore to show the lemma, all that is required is to show that it is suboptimal to turn a server on the moment another completes its setup. Consider a system in some valid state (i, j, m) where $j > 0$ and $i + m < C$.

Now assume that it just moved to state $(i+1, j, m-1)$ and the system immediately starts a setup moving it to state $(i+1, j, m)$. Also assume that this is a better choice than doing nothing. If this were not a better choice, then this case would not need to be considered. Now consider the system a moment before the server turned on (in state (i, j, m)). If the server were employing a policy which turned on the next server once the next setup completes, the energy cost the system incurs for the next two setups is expected to be

$$(mE_{Setup})/(m\gamma) + (mE_{Setup})/(m\gamma) = 2E_{Setup}/\gamma.$$

However, now consider the case (during the same moment before the setup completes) where the system now decides to begin turning the next server on immediately. The cost until the next two setups completes is now:

$$\frac{(m+1)E_{Setup}}{(m+1)\gamma} + \frac{mE_{Setup}}{m\gamma} = \frac{2E_{Setup}}{\gamma}.$$

Therefore, the setup cost of the next two servers to turn on (which is assumed to be beneficial to the system) is equal if the choice is made the moment before or after the first setup completes. However, the expected time for the servers to complete their setup is now less since the expected time for the first server to complete its setup is reduced by:

$$\frac{1}{m\gamma} - \frac{1}{(m+1)\gamma}.$$

Therefore the total cost is less when choosing to turn a server on before the setup

completes. Due to the Markovian nature of the system, this implies that it would have been better to make this decision at the most recent event. But due to the previous observations regarding the suboptimal nature of making setup decisions during departures and server turn offs, it is known it would have been better to begin turning the server on at the last arrival. □

Lemma A.2. *For all* i, m *such that* $0 \leq i < (m+1) \leq C$, $k^+_{m+1,i} \leq k^+_{m,i+1}$.

Proof. This is proven via contradiction. Assume $k^+_{m,i+1} > k^+_{m+1,i}$. This would create a case where in state $(i, k^+_{m,i}, m-i)$ a server could complete its setup, and move the system to state $(i+1, k^+_{m,i}, m-i-1)$. Hence, because it is assumed $k^+_{m,i+1} > k^+_{m+1,i}$, another server will begin its setup. However, this violates Lemma A.1, therefore, the assumption is false. □

Lemma A.3. *For all* $0 < i < C$ *and* $0 \leq m < (C-i)$, $k^-_{i,m+1} \leq k^-_{i+1,m}$.

Proof. Consider two systems S_1 and S_2 in valid states $(i, j, m+1)$ and $(i+1, j, m)$, respectively. Now assume that in S_1 it is optimal turn a server off. That is, the cost of having the ith server on (energy, future state of other servers, etc.), is greater than the cost of turning it off (holding cost, future state of other servers, etc.). Due to this assumption, if it can be shown that: if by turning off a server in S_2 it can incur a lesser cost than S_1, then it must also hold that it is optimal to turn a server off in S_2. Notice that S_2 can mimic S_1's behaviour by doing the following. S_2 turns off the ith server and lets the $(i+1)$th server idle. For the remainder of the proof, the $(i+1)$th server in S_1 is referred to as s_1, and the $(i+1)$th server in S_2 is referred to as s_2.

Notice that there is exactly one more server in setup in S_1 than in S_2. Due to the server homogeneity, this extra server in setup can be thought of as s_1. From here

one of two things can happen: s_1 either completes its setup process, or has its setup cancelled. When s_1 completes its setup, S_2 forces s_2 to mimic the behaviour of s_1 (idling, busy, etc.) and S_1 and S_2 are synchronized. Moreover, until S_1 and S_2 become synchronized, the only difference in costs are those contributed by s_1 and s_2 (all other servers are synchronized and no extra jobs arrive or depart). Therefore, because the cost of idling s_2 is less than the cost of having s_1 in setup ($E_{Setup} > E_{Idle}$), by turning the initial ith server off in S_2, it can save even more cost than S_1 does by turning off the same server. And because it is assumed turning off a server is optimal in S_1, it is also optimal to turn a server off in S_2. In others words, $k^-_{i,m+1} \leq k^-_{i+1,m}$. □

Lemma A.4. *For all* $0 < i \leq C$ *and* $0 \leq j < C - i$, *and* $i + j \leq (C - 1)$, $k^-_{i,j} \leq k^-_{i,j+1}$.

Proof. The proof of this lemma is similar to that of Lemma A.3. That is, if it is optimal to turn a server off in valid state (i, j, m), a system in valid state $(i, j, m + 1)$ could mimic the behaviour while incurring a lesser cost. Furthermore, since turning a server off in (i, j, m) is optimal, it is also optimal in $(i, j, m+1)$. Therefore $k^-_{i,m} \leq k^-_{i,m+1}$. □

Taking Lemma A.2, Lemma A.3, and Lemma A.4 together, Theorem 5.4 follows immediately.

A.5 Proof of Theorem 5.5

For simplicity of navigation Theorem 5.5 is restated.

Theorem 5.5. *For all energy-aware systems, for all linear well-formed cost functions, if the number of jobs in the system is greater than or equal to the number of servers currently turned on, it is always suboptimal to turn a server off, which is already on.*

Proof. The proof of this theorem is achieved via noting a contradiction in the optimality equations of the MDP if the negation of the theorem is assumed to be true. Note, that a linear well-formed cost function, i.e. $\mathbb{E}[R] + \beta\mathbb{E}[E]$, is minimized when one employs the optimal policy determined by the immediate cost function

$$w((i,j),a) = j + \beta[\max(0,(i-j+\min(0,a)))E_{\text{Idle}} + \max(0,a)E_{\text{Setup}}].$$

As such, the above immediate cost function is assumed to be used for the remainder of this proof. Assume that $j \geq i$ (there are at least as many jobs in the system as there are servers on), and that it is optimal to turn a server off. From the optimality equations, that is true if and only if $u(i,j,-1) \leq u(i,j,0)$.

$$\begin{aligned}
& u(i,j,-1) \leq u(i,j,0) \\
\Leftrightarrow \quad & w((i,j),-1) \\
& \quad + \lambda u(i-1,j+1) + (i-1)\mu u(i-1,j-1) + [1-\lambda-(i-1)\mu]u(i-1,j) \\
& \quad \leq w((i,j),0) + \lambda u(i,j+1) + i\mu u(i,j-1) + [1-\lambda-i\mu]u(i,j) \\
\Leftrightarrow \quad & j + \lambda u(i-1,j+1) + (i-1)\mu u(i-1,j-1) + [1-\lambda-(i-1)\mu]u(i-1,j) \\
& \quad \leq j + \lambda u(i,j+1) + i\mu u(i,j-1) + [1-\lambda-i\mu]u(i,j) \\
\Leftrightarrow \quad & \lambda u(i-1,j+1) + (i-1)\mu u(i-1,j-1) + [1-\lambda-(i-1)\mu]u(i-1,j) \\
& \quad \leq \lambda u(i,j+1) + i\mu u(i,j-1) + [1-\lambda-i\mu]u(i,j) \qquad (3)
\end{aligned}$$

It is noted that for all n and m, $u(n,m) \leq u(n-1,m)$. This follows from noting that in state (n,m) the system could always immediately turn a server off bringing it to state $(n-1,m)$ without additional cost. Therefore, (3) implies

$$[1-\lambda-(i-1)\mu]u(i-1,j) \quad \leq \quad [1-\lambda-i\mu]u(i,j)+\mu u(i,j-1). \tag{4}$$

While it is true that for all n and m, $u(n,m) \leq u(n-1,m)$ something even stronger can be said regarding $u(i,j)$ and $u(i-1,j)$. Because it is assumed that it is better to turn a server off in state (i,j) and that servers can turn off instantly it is known $u(i,j) = u(i-1,j)$. Therefore, (4) implies

$$\begin{aligned}
[1-\lambda-(i-1)\mu]u(i,j) &\leq [1-\lambda-i\mu]u(i,j)+\mu u(i,j-1) \\
-(i-1)\mu u(i,j) &\leq -i\mu u(i,j)+\mu u(i,j-1) \\
\mu u(i,j) &\leq \mu u(i,j-1) \\
u(i,j) &\leq u(i,j-1).
\end{aligned}$$

But this is a clear contradiction since it cannot be the case that a system with more jobs in it can do just as well as a system with less jobs given all other things are equal. For example, consider the sample path of the ith server idling with $j-1$ jobs in one system, while the ith server processes the jth job in the other.

□

A.6 Proof of Theorem 5.6

For ease of reading Theorem 5.6 is restated.

Theorem 5.6. *For all energy-aware systems, for all linear well-formed cost functions, the optimal policy turns servers on following a bulk setup scheme.*

Proof. As noted previously, a linear well-formed cost function, i.e. $\mathbb{E}[R] + \beta\mathbb{E}[E]$, is minimized when one employs the optimal policy determined by the immediate cost function

$$w((i,j),a) = j + \beta[\max(0,(i-j+\min(0,a)))E_{\text{Idle}} + \max(0,a)E_{\text{Setup}}].$$

As such, the above immediate cost function is assumed to be used for the remainder of this proof. The strategy is to leverage the optimality equations of the MDP to show the bulk setup nature of the optimal policy. It is assumed that in state (i,j) it is better to have a servers in setup, where $a > 0$, rather than choose to do nothing. The following inequality is then known from the optimality equations.

$$u(i,j,a) \leq u(i,j,0)$$

$$\begin{aligned}
\Leftrightarrow \quad & w((i,j),a) + \lambda u(i,j+1) + a\gamma u(i+1,j) + \min(i,j)\mu u(i,j-1) \\
& + [1-\lambda-a\gamma-\min(i,j)\mu]u(i,j) \\
& \leq w((i,j),0) + \lambda u(i,j+1) + \min(i,j)\mu u(i,j-1) + [1-\lambda-\min(i,j)\mu]u(i,j)
\end{aligned}$$

$$\Leftrightarrow \quad a\beta E_{\text{Setup}} + a\gamma u(i+1,j) - a\gamma u(i,j) \leq 0$$

$$\Leftrightarrow \quad \frac{\beta E_{\text{Setup}}}{\gamma} \leq u(i,j) - u(i+1,j)$$

This derivation can be expressed more succinctly as

$$u(i,j,a) \leq u(i,j,0) \quad \Leftrightarrow \quad \frac{\beta E_{\text{Setup}}}{\gamma} \leq [u(i,j) - u(i+1,j)], \tag{5}$$

for all valid actions $a \geq 0$. Now consider the action of turning on all available servers. If it can be shown that this option is at least as good as turning on an arbitrary choice of a (when $a \geq 0$), then the proof is complete. This is expressed mathematically as follows.

$$u(i,j,C-i) \leq u(i,j,a)$$

$$\begin{aligned}
\Leftrightarrow \quad & w(i,j,C-i) + \lambda u(i,j+1) + (C-i)\gamma u(i+1,j) + \min(i,j)\mu u(i,j-1) \\
& + [1 - \lambda - (C-i)\gamma - \min(i,j)\mu]u(i,j) \\
& \leq w(i,j,a) + \lambda u(i,j+1) + a\gamma u(i+1,j) + \min(i,j)\mu u(i,j-1) \\
& + [1 - \lambda - a\gamma - \min(i,j)\mu]u(i,j)
\end{aligned}$$

$$\Leftrightarrow \quad (C-i-a)\beta E_{\text{Setup}} + (C-i-a)\gamma u(i+1,j) \leq (C-i-a)\gamma u(i,j)$$

$$\Leftrightarrow \quad \frac{\beta E_{\text{Setup}}}{\gamma} \leq u(i,j) - u(i+1,j)$$

This derivation can be expressed more succinctly as

$$u(i,j,C-i) \leq u(i,j,a) \quad \Leftrightarrow \quad \frac{\beta E_{\text{Setup}}}{\gamma} \leq [u(i,j) - u(i+1,j)],$$

for all valid actions a. Then from (5)

$$u(i,j,C-i) \leq u(i,j,a) \quad \Leftrightarrow \quad u(i,j,a) \leq u(i,j,0),$$

for all valid actions $a \geq 0$. In other words, if it is optimal to turn at least one server on, then it is optimal to turn all available servers on. Hence it is optimal to turn servers on following a *bulk setup* scheme.

□

B Unused Formulations and Lemmas

As is often the case with research, some results although interesting, become less valuable as later observations are made. Here this work presents results which were used in previous versions of the proofs found in Appendix A, but are no longer. However, the reader or future researcher may find them valuable. As such, they are given in full detail.

In addition to the traditional optimality equations (1) and (4), it is useful to also define a set of *constrained optimality equations* defined as

$$\begin{aligned} w^* + u(i,j,m,a) = \mathcal{C}(i,j,m,a) &+ \lambda u(i+\min(0,a), j+1, m+\max(0,a)) \\ &+ (m+\max(0,a))\gamma u(i+\min(0,a)+1, j, m+\max(0,a)-1) \\ &+ \min(j,(i+\min(0,a+m)))\mu u(i+\min(0,a), j-1, m+\max(0,a)) \\ &+ [1-\lambda-(m+\max(0,a))\gamma-(i+\min(0,a))\mu] \\ &\quad \cdot u(i+\min(0,a), j, m+\max(0,a)), \end{aligned}$$

for the MDP given in (5.3), and

$$\begin{aligned} w^* + u(i,j,a) = w((i,j),a) &+ \lambda u(i + \min(0,a), j+1) + \max(0,a)\gamma u(i+1,j) \\ &+ \min(j, (i + \min(0,a)))\mu u(i + \min(0,a), j-1) \\ &+ (1 - \lambda - \max(0,a)\gamma - (i + \min(0,a))\mu) u(i + \min(0,a), j). \end{aligned}$$

These equations follow the optimal actions (determined by the traditional optimality equations), except at the first decision epoch, where the action is explicitly specified as a. It will be seen that a proof of one of the lemmas aided through the use of these equations.

While the above formulations are powerful in their own right, they deal with infinite recursions and can be cumbersome for certain properties. Another tool can be employed however, which allows for reasoning about policies over finite time horizons (and finite cost), while still retaining the optimal policy of the infinite time horizon case. These corresponding finite costs are built around the renewal reward argument. Let $c(s)$ be the minimum ratio over all possible policies of the expected cost incurred during a cycle of leaving and returning to state s and the expected time it takes to complete that cycle. Formally this is defined as follows. Let $\mathcal{R}_{\pi,s}$ be a random variable denoting the reward (or cost) earned over a single cycle of state s under policy π. A cycle of state s refers to the interval of time where the system begins in state s, leaves that state, and then returns to it at some point in the future, which completes the cycle. Furthermore, let $T_{\pi,s}$ be a random variable denoting the amount of time it

takes to complete a cycle of s under policy π. Given the set of all policies Π, along with these two random variables, $c(s)$ can be defined as

$$c(s) = \min_{\pi \in \Pi} \left\{ \frac{\mathbb{E}[\mathcal{R}_{\pi,s}]}{\mathbb{E}[T_{\pi,s}]} \right\}.$$

Here $\mathcal{R}_{\pi,s}$ is interpreted as a cost, since the minimum is used. This is a convenient definition since the renewal reward theorem can be applied. That is, if $\mathcal{R}_\pi(t)$ denotes the total reward/cost earned by time t under policy π, then for all states s

$$\lim_{t \to \infty} \frac{\mathcal{R}_\pi(t)}{t} = \frac{\mathbb{E}[\mathcal{R}_{\pi,s}]}{\mathbb{E}[T_{\pi,s}]}.$$

This gives the interpretation that $c(s)$ is the rate at which the system gains reward/cost under the optimal policy. This is true no matter which state s is chosen as the cycle reference, or as it is referred to in the literature, as the renewal process [28]. Therefore $c(s)$ is independent from s, in other words, $c(s)$ is a constant, and will be denoted by c^*. Likewise, the ratio $\mathbb{E}[\mathcal{R}_{\pi,s}]/\mathbb{E}[T_{\pi,s}]$ is also independent from s and an arbitrary state, say $s = (C_S, 0, 0)$ can be chosen as the cycle reference. It should be noted that this does not say $\mathbb{E}[\mathcal{R}_{\pi,s}]$ and $\mathbb{E}[T_{\pi,s}]$ do not depend on s, since they clearly do, but rather that their ratio is independent of s. Therefore for all s and some specific policy π, $\mathbb{E}[\mathcal{R}_{\pi,s}]/\mathbb{E}[T_{\pi,s}]$ equals a constant, denoted by c_π.

Similar to the constrained optimality equations, an extension $c(s, a)$ is made to c^*. The quantity $c(s, a)$ is no longer the minimum ratio of the expected cost incurred and expected cycle time over all policies, but rather the minimum ratio over all policies which choose action a in state s. Formally, given the set of all stable policies which

choose action a in state s, denoted by $\Pi_{s,a}$,

$$c(s,a) = \min_{\pi \in \Pi_{s,a}} \{c_\pi\}.$$

The values $c(s,a)$ are referred to as the constrained cycle costs. Such a construction is useful for several reasons, but perhaps the most important is that this construct allows one to reason about cost functions which would not be feasible within the MDP framework. For example, a popular cost function is the ERP, i.e. $\mathbb{E}[E]\mathbb{E}[R]$. This cost function cannot be used in an MDP, since the MDP is constrained to linear cost functions, i.e. $\mathbb{E}[R] + \beta\mathbb{E}[E]$. However, from renewal analysis, it is known that there must exist a random variable $\mathcal{R}_{\pi,s}$ which denotes some reward gained over a renewal cycle of state s under policy π, such that:

$$\min_{\pi \in \Pi} \left\{ \mathbb{E}[E^\pi]\mathbb{E}[R^\pi] \right\} = \min_{\pi \in \Pi} \left\{ \frac{\mathbb{E}[\mathcal{R}_{\pi,s}]}{\mathbb{E}[T_{\pi,s}]} \right\},$$

where E^π and R^π are random variables which denote the energy consumed and response time under policy π, respectively.

Lemma B.1. *For all valid states s and for all actions $a \in A_s$, a is an optimal action in state s if and only if $c(s,a) = c^*$.*

Proof. This result is proven via contradiction. Assume $c(s,a) = c^*$, but a is a suboptimal action. That is, for all optimal policies π^*, π^* is not present in the constrained set of policies $\Pi_{s,a}$. Therefore,

$$c(s,a) = \min_{\pi \in \Pi_{s,a}} \{c_\pi\} > c_{\pi^*} = c^*$$

which implies $c(s,a) > c^*$ and contradicts the initial assumption. □

Lemma B.2. *Let $S_1 = (C, \lambda, \mu, \gamma_1)$ and $S_2 = (C, \lambda, \mu, \gamma_2)$ be such that $\gamma_1 > \gamma_2$ and $c_1(\cdot)$ and $c_2(\cdot)$ denote the constrained cycle costs for S_1 and S_2 respectively. Then for all valid states s and actions a, $c_1(s,a) < c_2(s,a)$ for all well-formed cost functions.*

Proof. Let $\widetilde{S}_1 = (C, \lambda, \mu, \gamma_1)$ and $\widetilde{S}_2 = (C, \lambda, \mu, \gamma_2)$ with the condition that they also have identical probability spaces. That is, all interarrival and service times are equal, and the setup times are generated from the same random variable. In other words, letting random variables $\Gamma_{1,i}$ and $\Gamma_{2,i}$ denote the ith setup time in $\widetilde{S}_1$ and $\widetilde{S}_2$ respectively, $\Gamma_{1,i} = X_i/\gamma_1$ and $\Gamma_{2,i} = X_i/\gamma_2$, where $X_i \sim \exp(1)$. Using the methodology in [62], and noting that S_i and $\widetilde{S}_i$ have the same underlying distributions, if it can be shown that for any policy $\widetilde{S}_2$ employs, $\widetilde{S}_1$ can always do better with regards to the metrics $\mathbb{E}[R]$ and $\mathbb{E}[E]$, then $c_1(s,a) < c_2(s,a)$ for all well-formed cost functions. Showing that $\widetilde{S}_1$ can always do better than $\widetilde{S}_2$ is straightforward. Assume $\widetilde{S}_1$ mimics the behaviour of $\widetilde{S}_2$ in every way with the following exception: when the ith server in $\widetilde{S}_1$ completes its setup before the ith server in $\widetilde{S}_2$ (which it is guaranteed to do), it remains idle until the setup of the ith server in $\widetilde{S}_2$ either completes its setup or is cancelled, at which point the ith server in $\widetilde{S}_1$ again begins to mimic its behaviour (by either turning off, processing a job, or remaining idle). During this interval where the servers are unsynchronized $\widetilde{S}_1$ incurs a lesser energy cost since $E_{\text{Idle}} < E_{\text{Setup}}$. Therefore for all well-formed cost functions $c_1(s,a) < c_2(s,a)$. □

Lemma B.3. *Given an energy-aware system, $S = (C, \lambda, \mu, -)$, where the absence of a setup rate denotes instant setup times, for all well-formed cost functions, it is optimal to have the number of servers on be equal to the number of jobs in the system, whenever possible. Formally, in state (i,j), $a = \min(C,j) - i$.*

Proof. This policy is shown to simultaneously minimize the expected energy cost and the expected response time. The lowest energy cost which can be incurred is the energy required to process all the jobs. This is the energy cost incurred in the policy where $a = \min(C, j) - i$, since here a server is never idle, or in setup (as setup times are 0). Therefore, under the proposed policy, the expected energy costs are minimized.

It is a similar story with the expected response time. It is known that the mean response time is minimized when jobs are either served immediately if the number of jobs in the system is less than C, and a job is waiting in the queue if and only if all C servers are busy processing other jobs. This is also true for the action $a = \min(C, j)-i$, since due to the instant turn on times, jobs never wait in the queue, unless all servers are currently busy.

□

Lemma B.4. *Given two energy-aware systems $S_1 = (C, \lambda, \mu, \gamma)$ and $S_2 = (C, \lambda, \mu, -)$, where the absence of a setup rate denotes instant setup times, and where π^* is an optimal policy for S_2, if $\mathcal{C}_i^\pi$ denotes some cost for S_i employing policy π, then as $\gamma \to \infty$, $\mathcal{C}_1^{\pi^*} \to \mathcal{C}_2^{\pi^*}$.*

Proof. Consider two energy-aware systems $S_1 = (C, \lambda, \mu, \gamma_1)$, where $1/\gamma_1 = \epsilon$ and $S_2 = (C, \lambda, \mu, -)$, where the absence of a setup rate denotes instant setup times. Let π^* be the policy described in Lemma B.3. From Lemma B.3 it is known that π^* is an optimal policy for S_2. It will be shown that as $\epsilon \to 0$, π^* is also an optimal policy for S_1.

Firstly, it must be shown that the metrics which make up the cost function, holding

costs and energy costs, are continuous in ϵ, specifically around 0. This is done by looking directly at the expected reward/cost and cycle time for the cycle equations of S_1 under π^* and how they compare to the corresponding values in S_2.

Since at the moment, individual metrics are of concern, the reward/cost function will be defined as one of these metrics. Firstly the rate of energy consumption is considered. Letting the cost function simply be the energy metric, by definition:

$$c_2^* = \frac{\mathbb{E}[E_{2,\pi^*,s}]}{\mathbb{E}[T_{2,\pi^*,s}]},$$

where the subscript 2 denotes the values corresponding to S_2, E is a random variable denoting the total energy cost across the cycle of s, $s = (0, C_S, 0)$ and C_S denotes the number of servers which always remain on. Now if it can be shown that as $\epsilon \to 0$, $\mathbb{E}[E_{1,\pi^*,s}] \to \mathbb{E}[E_{2,\pi^*,s}]$ and $\mathbb{E}[T_{1,\pi^*,s}] \to \mathbb{E}[T_{2,\pi^*,s}]$ then the expected rate of energy consumption is continuous in the expected setup time around 0. Looking at the energy consumed over a single cycle one can bound the expected energy consumed by S_1 below and above.

$$\begin{aligned}\mathbb{E}[E_{2,\pi^*,s}] \leq \mathbb{E}[E_{1,\pi^*,s}] \leq \\ \mathbb{E}[E_{2,\pi^*,s}] + \epsilon\mathbb{E}[N_s]E_{\text{setup}} + (1 - \mathbb{E}[P(N_s)])\mathbb{E}[\mathcal{C}_E(N_s)|X = 1] \\ + \mathbb{E}[P(N_s)]\mathbb{E}[\mathcal{C}_E(N_s)|X = 0],\end{aligned} \tag{6}$$

where N_s is a random variable denoting the number of server setups before completing a cycle of state s in S_2. The third and fourth terms require a little more explanation. Because there are turn on times in S_1, the expected extra amount of time it takes to

turn the servers on not only adds directly, but could cause another job to arrive to the system before it returns to state $(0, C_S, 0)$. These terms represent the additional cost incurred in this case. While there could be servers setting up concurrently, in the worst case the cycle time is increased by N_s consecutive setups. However, if the N_s setups were all able to complete before a new job arrives to the system, this case of causing a potential new cycle to start would be avoided. The random variable X is an indicator variable which equals one if no new cycle is caused, and zero otherwise. $P(n)$ is a function which equals the probability of n consecutive server setups finishing before a new job arrives. Explicitly,

$$P(n) = \left(\frac{1/\epsilon}{1/\epsilon + \lambda}\right)^n.$$

One minus $P(n)$ represents the probability that a job would arrive before these setups finish. In other words, S_1 does not complete the cycle where S_2 would have. $\mathcal{C}_E(N_s)$ denotes all the added energy costs that starting a new cycle would add given there were N_s setups, including any consequent cycles which the new cycle may cause to occur. However, since the change in setup times has no impact on stability, $\mathcal{C}_E(N_s)$ must be finite.

There are a few things to note about (6). Firstly, given that no extra cycles occur, clearly the expected additional extra energy cost is equal to zero, i.e.

$$\mathbb{E}[P(N_s)]\mathbb{E}[\mathcal{C}_E(N_s)|X = 0] = 0.$$

Secondly, $P(n)$ is convex in n. Therefore, when applied to a random variable such as

N_s, from Jensen's inequality it is known that,

$$P(\mathbb{E}[N_s]) \leq \mathbb{E}[P(N_s)] \Rightarrow (1 - \mathbb{E}[P(N_s)]) \leq (1 - P(\mathbb{E}[N_s]))$$

Due to this inequality, (6) becomes,

$$\mathbb{E}[E_{2,\pi^*,s}] \leq \mathbb{E}[E_{1,\pi^*,s}] \leq \mathbb{E}[E_{2,\pi^*,s}] + \epsilon\mathbb{E}[N_s]E_{\text{Setup}} + \left(1 - \frac{1/\epsilon}{1/\epsilon + \lambda}\right)^{\mathbb{E}[N_s]} \mathbb{E}[\mathcal{C}_E(N_s)]. \tag{7}$$

From inspection of (7) it can be seen that as $\epsilon \to 0$, $\mathbb{E}[E_{1,\pi^*,s}] \to \mathbb{E}[E_{2,\pi^*,s}]$.

To show that the expected rate of energy consumption is continuous in the expected setup times around 0, it remains to show that as $\epsilon \to 0$, $\mathbb{E}[T_{1,\pi^*,s}] \to \mathbb{E}[T_{2,\pi^*,s}]$. The argument for this is very similar to that used when dealing with the expected energy consumption rate. Again bounds are developed, with the previous observations on (6) already applied.

$$\mathbb{E}[T_{2,\pi^*,s}] \leq \mathbb{E}[T_{1,\pi^*,s}] \leq \mathbb{E}[T_{2,\pi^*,s}] + \epsilon\mathbb{E}[N_s] + \left(1 - \frac{1/\epsilon}{1/\epsilon + \lambda}\right)^{\mathbb{E}[N_s]} \mathbb{E}[\mathcal{C}_T(N_s)], \tag{8}$$

where $\mathcal{C}_T(N_s)$ denotes the extra time added if S_1 were to fail completing a cycle due to the added setup times, where S_2 would not, given N_s setups occurred. Again this term is finite due to the stability of the system. From inspection of (8), as $\epsilon \to 0$, $\mathbb{E}[T_{1,\pi^*,s}] \to \mathbb{E}[T_{2,\pi^*,s}]$. Therefore, if the cost function is total energy cost, then as $\epsilon \to 0$, $c_1^* \to c_2^*$, and π^* is the optimal policy for S_1.

Since all well-formed cost functions are composed of the expected response time, as well as expected energy cost, to show that all cost functions are continuous in ϵ around 0, all that remains to show is that the holding costs are also continuous in ϵ around 0. The holding cost being continuous implies the mean number of jobs is continuous, which implies the mean response time is continuous. Here it is assumed that the cost function is simply the holding cost. Then by definition:

$$c_2^* = \frac{\mathbb{E}[H_{2,\pi^*,s}]}{\mathbb{E}[T_{2,\pi^*,s}]},$$

where H is a random variable denoting the holding cost incurred over a cycle of s. Using the previous observation on (6), bounds for the expected holding costs can be developed.

$$\begin{aligned}
&\mathbb{E}[H_{2,\pi^*,s}] \leq \mathbb{E}[H_{1,\pi^*,s}] \\
&\leq \mathbb{E}[H_{2,\pi^*,s}] + \epsilon\mathbb{E}[N_a] + \left(1 - \frac{1/\epsilon}{1/\epsilon + \lambda}\right)^{\mathbb{E}[N_s]} \mathbb{E}[\mathcal{C}_H(N_s)], \qquad (9)
\end{aligned}$$

where N_a is a random variable denoting the number of jobs which arrive during a cycle, and $\mathcal{C}_H(N_s)$ denotes the amount of holding cost incurred if a cycle is not completed in S_1 where it otherwise would have been in S_2, given N_s setups occurred. Similar to previous arguments, $\mathcal{C}_H$ is finite. From inspection of (9), as $\epsilon \to 0$, $\mathbb{E}[H_{1,\pi^*,s}] \to \mathbb{E}[H_{2,\pi^*,s}]$. Therefore, if the cost function is the rate at which holding costs are incurred, then as $\epsilon \to 0$, $c_1^* \to c_2^*$, and π^* is the optimal policy for S_1. Putting this together with the previous result that π^* is the optimal policy for the expected energy cost, it can be concluded that for S_1, as $\epsilon \to 0$, π^* is the optimal policy for all well-formed cost functions. □

C Proofs of Theorems of Chapter 7

Here the proofs of Theorems 7.1 and 7.2 are presented in detail. Before this can be done however, some preliminary material, which is used throughout the proofs, must first be defined and presented. Consider the following three sequences of systems with $0 < \rho < 1$:

1. Let S_1 be a sequence of n energy-aware queueing systems, where the ith energy-aware queueing system is given by $S_{1,i} = (C_{1,i}, \lambda_{1,i}, \mu_{1,i}, \gamma_{1,i})$, which employs some policy $\pi_i \in \Pi_A$, where ($\forall i \; s.t. \; 0 < i \leq n : \mu_{1,i} = 1$, $C_{1,i} = i$, and $\lambda_{1,i}/C_{1,i} = \rho$), and as $n \to \infty$, $\lambda_{1,n} \to \infty$.

2. Let S_2 be a sequence of n $M/M/C$ queues, where the ith $M/M/C$ queue is denoted by $S_{2,i} = (C_{2,i}, \lambda_{2,i}, \mu_{2,i})$, where ($\forall i \; s.t. \; 0 < i \leq n : \mu_{2,i} = 1$, $C_{2,i} = i$, and $\lambda_{2,i}/C_{2,i} = \rho$), and as $n \to \infty$, $\lambda_{2,n} \to \infty$.

3. Let S_3 be a sequence of n $M/M/C$ queues, where the ith $M/M/C$ queue is denoted by $S_{3,i} = (C_{3,i}, \lambda_{3,i}, \mu_{3,i})$, where ($\forall i \; s.t. \; 0 < i \leq n : \mu_{3,i} = 1$ and $C_{3,i} = \lambda_{3,i} + (\lambda_{3,i})^{0.5+\epsilon}$), where $0 < \epsilon < 0.5$, and as $n \to \infty$, $\lambda_{3,n} \to \infty$.

In order to compare and reason about these systems, let it also hold that: ($\forall i \; s.t. \; 0 < i \leq n : \lambda_{1,i} = \lambda_{2,i} = \lambda_{3,i} = \lambda_i$). Note that imposing such a constraint implies that ($\forall i \; s.t. \; 0 < i \leq n : C_{1,i} = C_{2,i} = C_i$). In other words, the arrival rates of the systems across all three sequences are equal, and the total number of servers in the systems of sequences 1 and 2 are also equal.

Let $B_{1,i}$, $B_{2,i}$, and $B_{3,i}$ denote the number of always busy servers in the ith system of sequence S_1, S_2, and S_3 respectively. This work now provides a Lemma which is key

to understanding the behaviours of these energy-aware systems under the asymptotic regime.

Lemma C.1. *Given a sequence of energy-aware systems where each system employs a policy $\pi \in \Pi_A$, C_n, λ_n, and μ_n denote the number of servers, arrival rate, and service rate of the nth system respectively, and as $n \to \infty$, $\lambda_n, C_n \to \infty$ while $\lambda_n/(\mu_n C_n)$ is fixed to some ρ, where $0 < \rho < 1$, it holds that for the number of always busy servers in the nth system denoted by B_n,*

$$\lim_{n\to\infty} \frac{B_n}{\lambda} = \frac{1}{\mu}.$$

Proof. Without loss of generality one can set $\mu = 1$. Therefore, to show Lemma C.1, it is equivalent to prove

$$\lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} = 1.$$

This is done via a sample path argument regarding the sequences S_1 and S_2. Consider the systems $S_{1,n}$ and $S_{2,n}$ as $n \to \infty$. At any point in time the number of servers currently available (on and idle) in $S_{1,n}$ is less than or equal to the number of servers available in $S_{2,n}$. This follows from the fact that $S_{1,n}$ may have some of its C_n servers off or in setup, while $S_{2,n}$ has C_n servers on at all times. Therefore, taking the same arrival stream and job sizes for both systems, the number of jobs in $S_{2,n}$ is less than or equal to the number of jobs in $S_{1,n}$. Therefore, if s is busy in $S_{2,n}$, then s has enough workload to also be busy in $S_{1,n}$, but may not be busy due to it being switched off or in setup. Therefore, if s is an always busy server in $S_{2,n}$, then s has enough workload to be always busy in $S_{1,n}$. However, as $S_{1,n}$ is employing a policy from Π_A, specifically, from the second condition in the definition of Class A policies it is known a server

will never turn off if there is work to do, and from the first condition a server will eventually turn on from the threshold scheme, it follows that almost surely the servers which can be always busy, will be always busy. That is to say, if s is an always busy server in $S_{2,n}$, then s is an always busy server in $S_{1,n}$. Therefore,

$$\lim_{n\to\infty} B_{1,n} \geq \lim_{n\to\infty} B_{2,n}. \tag{10}$$

Furthermore, it is known that

$$\lim_{n\to\infty} \frac{B_{2,n}}{\lambda_n} = 1.$$

This is shown via the following argument. Let $N_{2,i}(t)$ denote the number of jobs in the system $S_{2,i}$ at time t, and let a corresponding diffusion-scaled process be denoted by $\hat{N}_{2,n}(t)$, where

$$\hat{N}_{2,n}(t) = \frac{N_{2,n}(t) - n\rho}{\sqrt{n\rho}}.$$

From Theorem 4.1 in [35] it can be seen that as $n \to \infty$, $\hat{N}_{2,n}(t)$ weakly converges to an Ornstein-Uhlenbeck process, and therefore, at each time point t as $n \to \infty$, $\hat{N}_{2,n}(t)$ is normally distributed. After some elementary algebra,

$$N_{2,n}(t) = \hat{N}_{2,n}(t)\sqrt{\lambda_n} + \lambda_n \quad \Rightarrow \quad \frac{N_{2,n}(t)}{\lambda_n} = \frac{\hat{N}_{2,n}(t)}{\sqrt{\lambda_n}} + 1.$$

Because as $n \to \infty$, $\hat{N}_{2,n}(t)$ is normally distributed with finite mean and variance, $\lim_{n\to\infty} \hat{N}_{2,n}(t)/\sqrt{\lambda_n} = 0$. This immediately implies,

$$\frac{N_{2,n}(t)}{\lambda_n} = 1.$$

Moreover, one can say that as $n \to \infty$ if there are almost surely at least x jobs in the

system at all time points t, then as $n \to \infty$ there are at least x always busy servers at all time points t. Therefore,

$$\lim_{n\to\infty} \frac{N_n(t)}{\lambda_n} = 1 \quad \Rightarrow \quad \lim_{n\to\infty} \frac{B_{2,n}}{\lambda_n} \geq 1.$$

After realizing this property of S_2, one can begin examining the implication on the behaviour of S_1. Specifically, from (10) it is known,

$$\lim_{n\to\infty} \frac{B_{2,n}}{\lambda_n} \geq 1 \quad \Rightarrow \quad \lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} \geq 1.$$

Hence all that remains to show Lemma C.1 is to prove

$$\lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} \leq 1.$$

This follows immediately however, after the observation that

$$\lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} > 1$$

is true only if the arrival rate of the system is greater than λ_n, which is a direct contradiction to the system definition. Therefore,

$$\lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} \geq 1, \text{ alongside } \lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} \leq 1$$

implies

$$\lim_{n\to\infty} \frac{B_{1,n}}{\lambda_n} = 1.$$

□

C.1 Proof of Theorem 7.1

For simplicity of navigation, Theorem 7.1 is restated.

Theorem 7.1. *All policies in Π_A are asymptotically optimal with regards to expected response time. In other words, given an energy-aware system, for any $\pi_a \in \Pi_A$, as $\lambda, C \to \infty$ and $\lambda/\mu C$ is fixed to be ρ, where $0 < \rho < 1$, $\mathbb{E}[R^{\pi_a}] \to 1/\mu$.*

A high-level description of the proof is as follows. It is determined that if a job J is served by an always busy server in the system $S_{3,n}$ as $n \to \infty$, then the expected response time of job J approaches its expected service time. With this in mind, the system $S_{1,n}$ is compared to $S_{3,n}$ as $n \to \infty$. It becomes clear that the expected response time of these systems is dominated by jobs which are served by always busy servers. Moreover, the limits of the expected response time of these two systems are equal. Therefore, the expected response time of $S_{1,n}$ approaches the expected service time, as $n \to \infty$.

Proof. As with the proof of Lemma C.1, without loss of generality one can set $\mu = 1$. Therefore, to prove the theorem it is enough to show that $\lim_{n\to\infty} \mathbb{E}[R_{1,n}] = 1$, where $\mathbb{E}[R_{i,j}]$ denotes the expected response time corresponding to the system at the jth index of the ith sequence of systems. The equality of this limit is shown via a sample path argument regarding S_1 and S_3. Before this argument is made however, some properties of S_3 must be shown.

Consider the system $S_{3,n}$. From the square root staffing rule, Theorem 15.2 of [28], it is known that for a system with C servers, where $C = \lambda_n + c\sqrt{\lambda_n}$, the probability of queueing gets arbitrarily close to 0 as c gets large. Keeping in mind the definition of

S_3, specifically that $C_{3,i} = \lambda_i + (\lambda_i)^{0.5+\epsilon}$, it can be said that for all finite c,

$$\lim_{n\to\infty} \frac{c\sqrt{\lambda_n}}{\lambda_n^{0.5+\epsilon}} = \frac{c}{\lambda_n^{\epsilon}} = 0,$$

and therefore in the system $S_{3,n}$ as $n \to \infty$, $P(N_{3,n} > C) \to 0$, where $N_{3,n}$ is a random variable denoting the number of jobs in $S_{3,n}$, which implies,

$$\lim_{n\to\infty} \mathbb{E}[R_{3,n}] = \frac{1}{\mu_{3,n}} = 1. \tag{11}$$

Moreover, identical to the reasoning presented in Lemma C.1, one can conclude

$$\lim_{n\to\infty} \frac{B_{3,n}}{\lambda_n} = 1. \tag{12}$$

From here the server pool is split into two distinct conceptual sets. The first set consists of the always busy servers, and the second set consists of all remaining servers, i.e. the servers which spend some of their time idle. Let these two sets of servers be denoted by $A_{3,i}$ and $I_{3,i}$, respectively. Then the expected response time can be expressed as a sum of terms,

$$\mathbb{E}[R_{3,n}] = P_{A_{3,n}}\mathbb{E}[R^A_{3,n}] + P_{I_{3,n}}\mathbb{E}[R^I_{3,n}], \tag{13}$$

where $P_{A_{3,n}}$ and $P_{I_{3,n}}$ denote the probability of a job being served from set $A_{3,n}$ or $I_{3,n}$, respectively, and $\mathbb{E}[R^A_{3,n}]$ and $\mathbb{E}[R^I_{3,n}]$ denote the expected response times given that a job is served by a server from set $A_{3,n}$ or $I_{3,n}$, respectively. Due to the homogeneity of the servers, it is assumed that if there are c servers currently on and $i \leq c$ jobs in the system, then servers one through i will be the servers which are busy. From

this it can be noted that always busy servers, i.e. servers from set $A_{3,n}$, have serving priority over the others when jobs arrive. That is, it is known that the probability of a particular job being served by an always busy server is greater than or equal to choosing it randomly and uniformly from the entire pool. In other words,

$$P_{A_{3,n}} \geq \frac{|A_{3,n}|}{C_{3,n}} = \frac{|A_{3,n}|}{\lambda_n + \lambda_n^{0.5+\epsilon}}.$$

Furthermore, from the definition of $A_{3,n}$ it is known that $|A_{3,n}| = B_{3,n}$, which implies,

$$\lim_{n\to\infty} P_{A_{3,n}} \geq \lim_{n\to\infty} \frac{B_{3,n}}{\lambda_n + \lambda_n^{0.5+\epsilon}} = \lim_{n\to\infty} \frac{B_{3,n}}{\lambda_n} = 1.$$

Noting $P_{I_{3,n}} = 1 - P_{A_{3,n}}$ alongside (11) and (13) it becomes clear that,

$$\lim_{n\to\infty} \mathbb{E}[R_{3,n}] = \lim_{n\to\infty} \mathbb{E}[R_{3,n}^A] = 1.$$

With the limit of $\mathbb{E}[R_{3,n}^A]$ explicitly solved, the proof proceeds with a sample path argument involving S_1 and S_3. As $A_{3,n}$ and $I_{3,n}$ denote the sets of always busy and sometimes idle servers respectively for $S_{3,n}$, let $A_{1,n}$ and $I_{1,n}$ denote the corresponding always busy and sometimes idle sets for $S_{1,n}$.

It is worth noting that while calling $I_{3,n}$ a set of sometimes idle servers is apt, applying the same notion to $I_{1,n}$ is somewhat inappropriate. $I_{1,n}$ is the set of servers which are not always busy. These servers potentially could spend zero time idling, i.e. they immediately switch off when they complete a job and the queue is empty. However, for the purposes of the proof it is only required that these servers spend some portion

of time not busy.

Continuing with the sample path argument, consider systems $S_{1,n}$ and $S_{3,n}$, as $n \to \infty$, with identical arrival processes. Jobs are viewed as being marked to be served by a server belonging to the sets $A_{1,n}$ and $A_{3,n}$, or $I_{1,n}$ and $I_{3,n}$, without loss of generality. If a job arrives to $S_{3,n}$ and is served from the set $A_{3,n}$, then the corresponding job in $S_{1,n}$ will almost surely be served from the set $A_{1,n}$. This is the case by noting from (12) and Lemma C.1 that

$$\lim_{n\to\infty} \frac{B_{1,n}}{B_{3,n}} = 1. \tag{14}$$

Moreover, if a job arrives to $S_{3,n}$ and is served from the set $I_{3,n}$, then the corresponding job in $S_{1,n}$ will almost surely be served from the set $I_{1,n}$. Firstly, it must be seen that $I_{1,n}$ has the capacity to handle the load which $I_{3,n}$ has the capacity to handle. This follows immediately by noting that

$$\lim_{n\to\infty} \frac{|I_{3,n}|}{\lambda_n} = 0 \text{ while } \lim_{n\to\infty} \frac{|I_{1,n}|}{C} = (1-\rho),$$

which implies,

$$\lim_{n\to\infty} \frac{|I_{3,n}|}{|I_{1,n}|} = 0.$$

As was done with $S_{3,n}$, one can decompose the expected response time of $S_{1,n}$ into two distinct components as follows,

$$\mathbb{E}[R_{1,n}] = P_{A_{1,n}}\mathbb{E}[R^A_{1,n}] + P_{I_{1,n}}\mathbb{E}[R^I_{1,n}].$$

From here two important observations are made. Firstly because $S_{1,n}$ is a stable system $\mathbb{E}[R^I_{1,n}]$ is known to be finite. Secondly, because $S_{1,n}$ and $S_{3,n}$ have identical arrival processes, it can be said

$$\lim_{n\to\infty} P_{A_{1,n}} = \lim_{n\to\infty} P_{A_{3,n}} = 1$$

which alongside (14) implies,

$$\lim_{n\to\infty} \mathbb{E}[R^A_{1,n}] = \lim_{n\to\infty} \mathbb{E}[R^A_{3,n}] = 1.$$

Therefore,

$$\lim_{n\to\infty} \mathbb{E}[R_{1,n}] = \lim_{n\to\infty} \mathbb{E}[R^A_{1,n}] = 1.$$

□

C.2 Proof of Theorem 7.2

For simplicity of navigation, Theorem 7.2 is restated.

Theorem 7.2. *All policies in Π_B are asymptotically optimal with regards to expected energy cost. In other words, given an energy-aware system, for any $\pi_b \in \Pi_B$, as $\lambda, C \to \infty$ and $\lambda/\mu C$ is fixed to be ρ, where $0 < \rho < 1$, $\mathbb{E}[E^{\pi_b}]/\lambda \to \mathbb{E}[E^{J,\pi_b}] \to E_{Busy}/\mu$.*

A high-level description of the proof is as follows. The notion of the energy cost contributed by a single job is examined. Specifically, the energy cost of a single job is examined based on the criterion of it being served by an always busy server or not. Lemma C.2 gives an exact value of the energy cost of a single job assuming it was

served by an always busy server. Moreover, this is a minimum value. On the other hand, Lemma C.3 shows if a job is not served by an always busy server, the energy cost is finite. From there, similar to the procedure in the proof of Theorem 7.1, it becomes clear that the total expected energy cost is dominated by jobs which are served by always busy servers, and therefore is minimized.

Definition C.1. ***Energy Cost of a Job:*** *Let E^J be a random variable which denotes the energy cost contributed by a randomly chosen job J. There are four contributing factors to consider when determining E^J for some job J.*

1. *Each job J is responsible for the energy required to process it.*

2. *If a job J is the first job which server s serves after completing its setup process, then J is responsible for the entire cost of the setup process of s, as well as any idling costs of s until the next time s is switched off.*

3. *If a job J is responsible for causing a server setup which is canceled due to that job entering service before the setup process completes, then J is also responsible for the energy cost incurred by that setup.*

4. *The idling cost of servers which never turn off is divided evenly among all jobs that pass through the system. Furthermore, a special case is added to this factor, which is if some of the servers are always busy servers, i.e. some of the system parameters are approaching infinity, then the aforementioned idling cost is evenly distributed only across jobs which are served by always busy servers, rather than across all jobs which pass through the system.*

As mentioned in Section 6.5 and from the definition of E^J, it is clear that

$$\mathbb{E}[E] = \lambda \mathbb{E}[E^J]. \tag{15}$$

Therefore, when $\mathbb{E}[E^J]$ is minimized, $\mathbb{E}[E]$ is minimized.

Lemma C.2. *In an energy-aware queueing system employing a Class B policy, if a job J is served by an always busy server, then $\mathbb{E}[E^J] = E_{Busy}/\mu$.*

Proof. The proof of this Lemma is argued after several key observations. To prove the Lemma, it is equivalent to show that if J is served by an always busy server, then $\lim_{n\to\infty} \mathbb{E}[E^J_{1,n}] = E_{\text{Busy}}$ (recall $\mu_{1,n} = 1$ for all n). Moreover, for there to be any always busy servers present in the system, it is required that $n \to \infty$. Therefore, it will be argued that

$$\lim_{n\to\infty} \mathbb{E}[E^J_{1,n} \mid J \text{ was served by an always busy server}] = E_{\text{Busy}}.$$

To show the above equality, the four contributing factors to $\mathbb{E}[E^J_{1,n}]$, from Definition C.1, are addressed individually and summed.

1. It is known that each job J will eventually be served, by an always busy server or otherwise, and therefore J incurs an expected cost of $E_{\text{Busy}}/\mu_{1,i} = E_{\text{Busy}}$, for all i.

2. If it is known that J is served by an always busy server, then it is trivially known that it is not the first job to be processed after a server completes its setup process. Therefore it can be said that J incurs no cost from this contributing factor.

3. The third contributing factor takes a little more care, but can be shown to also incur no cost via a contradiction argument. Assume that some job J is responsible for causing a server to start its setup process. If this is the case however, it is known that such a server will almost surely never cancel its setup nor turn off from the structural property given in Theorem 5.3, which Class B policies adhere to. Therefore, it cannot be the case that J caused a setup which was then canceled. Moreover, if J did cause some server s to start a setup process, then s would be idle an infinite amount of time before turning off, a direct violation of Class B policies. Taking this all into account, it is clear that if J is served by an always busy server, then it incurs no cost from starting an eventually canceled setup.

4. Because $n \to \infty$, the special case of this factor is invoked, i.e. jobs which are served by always busy servers are responsible for the full cost of the always idle servers. From the definition of Class B policies it is known that the number of always idle servers is less than $(1 - \lambda_{1,n}/C_{1,n})C_{1,n}^{\alpha}$, where $0 \leq \alpha < 1$. Therefore, costs from these idle servers are incurred at some rate less than $(1-\rho)C_{1,n}^{\alpha}E_{\text{Idle}}$. Moreover, from the proof of Theorem 1, it is known that as $n \to \infty$, the probability of being served by an always busy server approaches 1. It then follows that the rate at which jobs are served by always busy servers approaches the arrival rate, $\lambda_{1,n}$, as $n \to \infty$. Therefore, letting the expected contributing cost from these idle servers per job be denoted by $\mathbb{E}[E^{J,I}]$ and from the definition of S_1 knowing that $\lambda_{1,n} = C_{1,n}/\rho$:

$$\mathbb{E}[E_{1,n}^{J,I}] = \lim_{n\to\infty} \frac{(1-\rho)C_{1,n}^{\alpha}E_{\text{Idle}}}{\lambda_{1,n}} = \lim_{n\to\infty} \rho(1-\rho)E_{\text{Idle}}\frac{C_{1,n}^{\alpha}}{C_{1,n}}$$

$$= \lim_{n\to\infty} \rho(1-\rho) E_{\text{Idle}} \frac{1}{C_{1,n}^{(1-\alpha)}} = 0.$$

Therefore, the only contributing factor to $\mathbb{E}[E_{1,n}^J]$ given that J has been served by an always busy server, is the cost of processing it, which implies

$$\lim_{n\to\infty} \mathbb{E}[E_{1,n}^J \mid J \text{ is served by an always busy server}] = E_{\text{Busy}}.$$

□

Lemma C.3. *In an energy-aware queueing system employing a Class B policy, if a job J is not served by an always busy server, then $\mathbb{E}[E^J]$ is finite.*

Proof. Similar to the proof of Lemma C.2, this proof iterates through the contributing factors of Definition C.1 to show that the expectation of each factor is finite, and therefore, the expectation of their sum is finite, i.e. $\mathbb{E}[E^J]$ is finite.

1. The cost directly incurred from processing a job is trivially known to be finite, since it is directly proportional to the service time of the job.

2. If it is known that a job J has been served by a server s which regularly completes its setup process, i.e. as $t \to \infty$, $0 < \mathbb{E}[X_{\text{Setup}}(s,t)] < 1$, two cases must be considered due to Definition 3. The first and simpler case being that J is not the first job to be served by s, after s completes a setup process. Here no costs will be incurred and are trivially finite. The second and more interesting case is that J is the first job to be served by s following a setup process. Here J is responsible for the setup costs incurred by s alongside the idling costs incurred by s until the next time it shuts off. From the definition of Class B policies it is

immediately known that the cost incurred by s from idling is finite. So all that remains is to show that the setup cost associated with s is also finite. While at first glance this may seem trivial since it is known that the expected setup time of a server is finite, this is not quite the case. The policy could be such that when a threshold is reached not only does the setup process of one server begin, but many, potentially even all servers begin their setup process, and the next one to complete its setup is chosen as the server to use, while the remainder of the servers terminate or cancel their setups. If s was the only server used in the setup process, then the expected cost is simply E_{Setup}/γ, which is finite. But since the setup times are exponentially distributed, for all m, where m is the number of servers used in the setup process of s, the cost incurred is expected to be $mE_{\text{Setup}}/m\gamma = E_{\text{Setup}}/\gamma$, which is finite. Therefore, the expected energy cost incurred from the contributing factor of being the first job to be served after a completed setup process is finite.

3. The last contributing factor to consider is the case of job J causing servers to start their setup processes, but have them be cancelled before they complete. This is similar to the previous case of the server(s) completing its setup process. If it is known that a server's setup process is interrupted before it completes, due to the underlying exponential distribution, it can be said that the expected cost incurred is less than E_{Setup}/γ. Moreover, if it is known that the setup processes of m servers are interrupted before any of them complete, due to the underlying exponential distribution it can be said that the total cost incurred is less than $mE_{\text{Setup}}/m\gamma = E_{\text{Setup}}/\gamma$. Therefore, the cost incurred by J from causing server setups which are eventually interrupted is finite.

4. Because there are always busy servers present in the system, from the definition of E^J it is trivially known that zero costs are incurred by this factor, which is finite.

All of the contributing terms of $\mathbb{E}[E^J]$ are finite, therefore $\mathbb{E}[E^J]$ is also finite. □

With the proof of the Lemmas complete, the proof of Theorem 7.2 is now provided.

Proof. To prove the theorem, it is equivalent to show $\lim_{n\to\infty} \mathbb{E}[E_{1,n}]/\lambda_n = E_{\text{Busy}}$ under the assumption that the systems of S_1 are employing Class B policies. From the definition of $\mathbb{E}[E^J]$, it is known $\mathbb{E}[E] = \lambda\mathbb{E}[E^J]$. Therefore,

$$\lim_{n\to\infty} \mathbb{E}[E_{1,n}]/\lambda_n = \lim_{n\to\infty} \mathbb{E}[E^J_{1,n}].$$

Furthermore, using the notation introduced in the proof of Theorem 7.1, it is also known that,

$$\begin{aligned}\lim_{n\to\infty} \mathbb{E}[E^J_{1,n}] = \lim_{n\to\infty} & P_{A_{1,n}}[E^J_{1,n} \mid J \text{ is served from } A_{1,n}] \\ & + P_{I_{1,n}}[E^J_{1,n} \mid J \text{ is served from } I_{1,n}].\end{aligned}$$

Leveraging past equalities allows one to simplify the above equation. From Lemma C.2 it is known

$$\lim_{n\to\infty} [E^J_{1,n} \mid J \text{ is served from } A_{1,n}] = E_{\text{Busy}}.$$

From Lemma C.2, $[E^J_{1,n} \mid J \text{ is served from } I_{1,n}]$ is finite, i.e. for some $L_n > 0$

$$\lim_{n\to\infty} [E^J_{1,n} \mid J \text{ is served from } I_{1,n}] = L_n$$

From the proof of Theorem 7.1, it is known,

$$\lim_{n\to\infty} P_{A_{1,n}} = 1 \quad \text{and} \quad \lim_{n\to\infty} P_{I_{1,n}} = 0.$$

Therefore,

$$\lim_{n\to\infty} \mathbb{E}[E^J_{1,n}] = E_{\text{Busy}}. \tag{16}$$

It is worth noting that E_{Busy} is a lower bound for $\mathbb{E}[E^J_{1,n}]$ since for the system to be stable the job must be processed. In other words, as $n \to \infty$, $\mathbb{E}[E^J_{1,n}]$ approaches its minimum value. Returning to (15),

$$\lim_{n\to\infty} \mathbb{E}[E_{1,n}]/\lambda_n = \lim_{n\to\infty} \mathbb{E}[E^J_{1,n}],$$

it becomes clear from (16) that,

$$\lim_{n\to\infty} \mathbb{E}[E_{1,n}]/\lambda_n = E_{\text{Busy}}.$$

Moreover, in system $S_{1,i}$, $\lambda_i E_{\text{Busy}}$ is a trivial lower bound for the expected energy cost, implying that as $n \to \infty$, $\mathbb{E}[E_{1,n}]$ is minimized. That is, the policy which $S_{1,n}$ is employing, which is a Class B policy, minimizes the expected energy cost as $n \to \infty$. □