Optimal Control of a Heterogeneous Two Server System with Consideration for Power and Performance

by

Jiazheng Li

A thesis presented to the
University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis we consider a system of two heterogeneous servers with a shared queue, and examine a scheduling policy for the optimal control of such a system. Previous results by Lin and Kumar [1] and Koole [2] found that a threshold policy, i.e., refraining from assigning a job to a slow server until a certain threshold has been exceed in the job queue, is optimal when seeking only to minimize the mean sojourn time of a job in the system. We build upon these results and generalise the analytical proof of the threshold policy's optimality to take into account power consumption as another performance metric, in the setting where the faster server is more efficient. We also obtain preliminary results for a setting where the slower server is more efficient, under the restriction of low arrival rates. We use experimental data from simulations to provide an assessment of the real world applicability of a threshold policy in this setting; a comparison between a threshold policy with optimal thresholds and a first-come-first-serve policy shows that it achieves a cost improvement of up to 29.19% over the naive policy.

Acknowledgements

I would like to thank Professors Shreyas Sundaram and Siddharth Garg for providing me with this opportunity for research, the guidance throughout the course of my studies, and ultimately the encouragement that allowed me to persevere.

Table of Contents

List of Figures

Chapter 1

Introduction

With the increasing ubiquity of mobile devices and web services in recent years, reducing power usage in computer systems has become an increasingly critical design objective. For mobile devices, battery capacity is often limited by constraints on the device's weight and physical dimensions, and reducing power consumption whenever possible is key to improving run time performance. On the other side of the spectrum, the advent of web services and cloud computing have necessitated large scale server farms with thousands of computing nodes to adequately meet consumer needs. Server farms are highly energy intensive, and the operating costs incurred by energy consumption and removal of the resultant heat can rival the cost of the server hardware itself [3]; reducing the power usage in computing nodes reduces operating costs, decreases the strain imposed on the electrical grid, and reduces the emission of air pollutants and greenhouse gases [4].

In order to increase energy efficiency in hardware, various researchers (such as Cao et al. [5]) have proposed the use of asymmetric multicore processors (AMP) which consists of a fast, high power core and a slow, low power core in order to optimize both performance

and energy usage based on the workload.

In this thesis we examine a strategy for the optimal control of an AMP having two cores, one of which is more energy efficient than the other in terms of mean energy consumed per job during service. The AMP is modelled as a heterogeneous queueing system as shown in Figure ??, consisting of two exponential servers with different service rates μ_1 and μ_2. Incoming jobs are modelled as a Poisson process with rate λ and stored in a common queue while waiting to be serviced. After each system event, which can be the arrival of a new job into the queue or the departure of a job which has completed service, an assignment decision is made by the scheduling policy to decide if a job in the queue should be assigned to a server for service.

Performance is characterized here by the mean response time of the system, defined as the mean duration a job spends in the system from arrival until departure. Mean energy consumption is determined by charging a cost per unit time (E_1 and E_2 for server 1 and 2 respectively) for each time unit a server spends servicing a job. In a real-world setting, the fixed cost corresponds to the increase in power consumption of a server when it is serving a job compared to when it is idle. The goal of this thesis is to present an optimal control policy which minimizes a weighted sum of the mean response time and mean energy consumption, where varying the weight provides solutions on the Pareto boundary of mean response time and mean energy consumption.

1.1 Related Work

Lin and Kumar [1] studied the optimal control of the system shown in Figure ?? with the goal of minimizing only the mean response time. They formulated a discrete-time problem by sampling the state of the system when a transition event occurs, i.e., when a job arrives

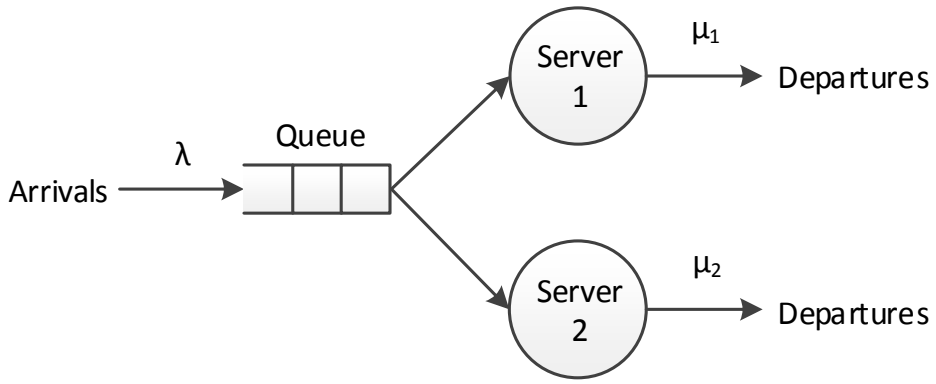

Figure 1.1: Queueing system model

at or departs from the system. The discrete-time problem is then modelled as a Markov decision process, and using value iteration they demonstrate that the faster server should be used whenever possible. Next, they use policy iteration to show that the optimal policy is a stationary policy of the threshold type, which keeps the faster server busy whenever possible and makes use of the slower server only if the number of jobs in the queue exceeds a certain threshold. The intuition behind this result is that when the arrival rate is low and not many jobs are in the queue, instead of sending a job to the slow server it may be better to wait for the fast server to finish service and service it there instead, resulting in a faster mean response time.

Building upon Lin and Kumar's result that the fast server should be kept busy whenever possible, Koole [2] simplifies the rest of the proof by using value iteration instead of policy iteration. He shows that over the infinite horizon, the difference in expected cost of a policy which does not send a job to the slow server compared to one that does is monotonically increasing with the queue length; when the difference is negative, it is better not to send jobs to the slow server, and as queue length increases, the difference eventually becomes

positive, at which point it becomes better to send jobs to the slow server. This demonstrates the optimality of threshold behaviour with the crossover point as the threshold value.

However, these proofs cannot be trivially extended to the setting where energy costs are taken into account. Rykov [6] made use of value iteration to show that the optimality of the threshold policy is preserved when additional queueing and service penalties are specified by using the minimum total average service cost as the criterion for server efficiency, under the restriction that more efficient servers also have higher service rates than less efficient servers; the case where the more efficient server is slower remains an open problem [7], and is one of the scenarios we consider in this thesis. We will make use of lower bounds on the cost difference between adjacent states to demonstrate that the optimal policy has characteristics of a threshold policy for the setting where the faster server is more efficient. In the setting where the slower server is more efficient, we will provide partial analytical results in conjunction with simulation results which demonstrate the cost benefit of the threshold policy compared to a naive first-come-first-serve policy.

Chapter 2

Discrete Time Problem Formulation

2.1 Discrete State Model

Adopting the strategy from [1], we begin by discretizing the continuous time system described in Chapter 1 by sampling the system at the instants in time when an event occurs, which can be a job arriving in the queue or leaving either server. In order to ensure the probability of sampling is the same regardless of system state, we must assume that if a server is idle then it is serving a "dummy" job, and therefore has the same probability of being sampled as if it were serving a real job. We also normalize λ, μ_1, and μ_2 such that $\lambda + \mu_1 + \mu_2 = 1$ while maintaining their proportion to each other, so that the probability of an event taking place, or equivalently, the probability of the system being sampled during the interval $(t, t + dt)$ is $(\lambda + \mu_1 + \mu_2)dt = dt$.

The state of the discretized system at time-step k is $x_k = (x_k^0, x_k^1, x_k^2) \in \mathbb{N} \times \{0, 1\} \times \{0, 1\}$, where x_k^0 is the number of jobs in the queue, and x_k^1 and x_k^2 are the number of jobs being serviced on server 1 and 2 respectively. Servicing a job on server 1 and 2 also incurs

a cost of $E_1 > 0$ and $E_2 > 0$ respectively.

Given the state of the system x_k at all time instants, the mean response time can be written as:

$$\bar{T} = \lim_{t\to\infty} \frac{1}{\lambda t} \sum_{k=1}^{t} \left(x_k^0 + x_k^1 + x_k^2\right),$$

and the mean cost can be written as:

$$\bar{E} = \lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{t} \left(E_1 x_k^1 + E_2 x_k^2\right).$$

The objective of the controller is to select control decisions which minimize $\bar{T} + \bar{\alpha}\bar{E}$, where $\bar{\alpha} > 0$ is a pre-specified weight factor.

In the rest of this paper, we will deal with the following equivalent objective function: $\lambda\bar{T} + \alpha\bar{E}$, where $\alpha = \lambda\bar{\alpha}$. By Little's Law [8], $\lambda\bar{T}$ is equal to the mean number of jobs in the system. As in the work by Lin and Kumar [1], we work with the discounted cost objective instead of the average cost objective:

$$E\left[\sum_{k=0}^{\infty} \beta^k g(x_k)\right], \tag{2.1}$$

where $\beta \in [0, 1)$ is a discount factor and

$$g(x_k) = x_k^0 + x_k^1 + x_k^2 + \alpha E_1 x_k^1 + \alpha E_2 x_k^2$$

is the stage cost at timestep k. Without loss of generality, we designate server 1 as the more efficient server, i.e.,

$$\frac{1 + \alpha E_1}{\mu_1} < \frac{1 + \alpha E_2}{\mu_2}. \tag{2.2}$$

Three system events are possible: an arrival into the system, a departure from server 1, or a departure from server 2. The state of the system after a new job arrival is indicated

by Ax_k and the state after a departure from server i is indicated by $D_i x_k$ ($i \in \{1, 2\}$); we define these states to be adjacent to x_k, and vice-versa. The state mappings are as follows:

$$A(x^0, x^1, x^2) = (x^0 + 1, x^1, x^2),$$
$$D_1(x^0, x^1, x^2) = (x^0, 0, x^2),$$
$$D_2(x^0, x^1, x^2) = (x^0, x^1, 0).$$

Note that the departure of a "dummy" customer does not change the state of the system.

After each event, one of the following four possible control decisions is selected for the transition to the next state: hold the job at the head of the queue (P_h), assign a job to server 1 (P_1), assign a job to server 2 (P_2), or assign jobs to both servers (P_b). The notation $x_k \in Dom(P_i)$ ($i \in \{0, 1\}$) is used to indicate that $x_k^0 > 0$ and $x_k^i = 0$, i.e., a job is available in the queue for assignment and server i is available to service it. The notation $x_k \in Dom(P_b)$ is used to indicate that $x_k^0 > 1$, $x_k^1 = 0$ and $x_k^2 = 0$, i.e., there are at least two jobs in the queue and both servers are free to begin service. The state of the system after applying control P_u is indicated by $P_u x_k$, where $u \in \{h, 1, 2, b\}$ and $x_k \in Dom(P_u)$. The state mappings are as follows:

$$P_h(x^0, x^1, x^2) = (x^0, x^1, x^2),$$
$$P_1(x^0, x^1, x^2) = (x^0 - 1, x^1 + 1, x^2) \text{ defined on } Dom(P_1),$$
$$P_2(x^0, x^1, x^2) = (x^0 - 1, x^1, x^2 + 1) \text{ defined on } Dom(P_2),$$
$$P_b(x^0, x^1, x^2) = (x^0 - 2, x^1 + 1, x^2 + 1) \text{ defined on } Dom(P_b).$$

2.2 Cost Modelling and Fixed Point Iteration

Let $J^\beta(x)$ represent the minimum cost over all policies with initial state x. The stochastic Bellman equation is

$$J^\beta(x) = g(x) + \beta\lambda \min_{u_0} J^\beta(P_{u_0}Ax) + \beta\mu_1 \min_{u_1} J^\beta(P_{u_1}D_1x) + \beta\mu_2 \min_{u_2} J^\beta(P_{u_2}D_2x).$$

Let $\mathcal{F}$ be the Banach space of all functions $f : \mathcal{X} \to \mathbb{R}$ where the norm $\|\cdot\|$ defined by

$$\|f\| = \sup_{x\in\mathcal{X}} \left| \frac{f(x)}{\max(g(x), 1)} \right|,$$

is finite, and define the dynamic programming operator $T : \mathcal{F} \to \mathcal{F}$ as

$$Tf(x) = g(x) + \beta\lambda \min_{u_0} f(P_{u_0}Ax) + \beta\mu_1 \min_{u_1} f(P_{u_1}D_1x) + \beta\mu_2 \min_{u_2} f(P_{u_2}D_2x).$$

Observation 1. *For some n, $T^{(n)}$ is a contraction mapping [9].*

By the Banach fixed-point theorem, for any $f \in \mathcal{F}$, $T^n f$ will converge to a unique fixed point w such that $Tw = w$ [10]. Since the optimal cost function $J^\beta \in \mathcal{F}$ cannot contract further, it stands to reason that $TJ^\beta = J^\beta$ and therefore J^β is the unique fixed point to which $\lim_{n\to\infty} T^n f$ converges. In the context of dynamic programming this technique is known as value iteration.

In the following chapter we will use value iteration to show that there is a non-empty and closed set of functions $\mathcal{G} \in \mathcal{F}$ having properties of a threshold policy and is invariant under T. In other words, for any $f \in \mathcal{G}$, Tf is also $\in \mathcal{G}$, therefore the fixed point $\lim_{n\to\infty} T^n f = J^\beta$ is in $\mathcal{G}$. By extension, since the fixed point is unique in $\mathcal{F}$, any $f \in \mathcal{F}$ will also converge to the same fixed point, and being in $\mathcal{G}$ we can conclude that it has properties of a threshold policy.

We note that when $\alpha = 0$, i.e., in the setting of [1], the stage costs satisfy

$$g(P_h x_k) = g(P_1 x_k) = g(P_2 x_k)$$

$$\text{for } x \in Dom(P_h) \cap Dom(P_1) \cap Dom(P_2).$$

The fact that the stage costs do not depend on the control decision simplifies the proof, since the the terms cancel out when comparing control actions. However when $\alpha > 0$, the proof becomes non-trivial as different control decisions incur different energy costs, and we must show that the difference in future costs over the infinite time horizon balances out the difference in immediate energy costs.

Chapter 3

Optimality of Threshold Policy When $\mu_1 > \mu_2$

In this chapter, we prove the optimality of threshold policies for the M/M/2 system described in Chapter 2, in the setting where the more power efficient server is also faster (i.e., $\mu_1 > \mu_2$ and $\frac{1+\alpha E_1}{\mu_1} < \frac{1+\alpha E_2}{\mu_2}$.). In Section 3.1 we provide expressions for lower bounds on the difference in costs between adjacent states. In Section 3.2 we use the results from Section 3.1 to show the optimal policy has the first characteristic of a threshold policy, i.e., the more efficient server should be kept busy whenever possible. Finally in Section 3.3 we use the results obtained in the previous sections to show that the optimal policy has the second characteristic of a threshold type policy, i.e., jobs should be sent to the less efficient server once the queue length exceeds some threshold. These two characteristics are sufficient to show that the optimal policy is of threshold type.

3.1 Bounds on Cost Difference Between Adjacent States

In this section, lower bounds are shown for the change in the optimal value function J^β immediately after a system event, which can be an arrival or a departure from either server. Note that the results in this section do not depend on $\mu_1 > \mu_2$, and are equally applicable to the setting where the slower server is more efficient.

Lemma 1. *There exists $\beta^* \in [0,1)$ such that for all $\beta \in [\beta^*, 1)$, the optimal value function J^β has the following properties for all x:*

1. $J^\beta(x^0+1, x^1, x^2) - J^\beta(x^0, x^1, x^2) \geq \delta_1$ *where*

$$\delta_1 = \min\left(\frac{1}{1-\beta}, \delta_2, \delta_3\right). \tag{3.1}$$

2. $J^\beta(x^0, 1, x^2) - J^\beta(x^0, 0, x^2) \geq \delta_2$ *where*

$$\delta_2 = \frac{1+\alpha E_1}{1-\beta(1-\mu_1)}. \tag{3.2}$$

3. $J^\beta(x^0, x^1, 1) - J^\beta(x^0, x^1, 0) \geq \delta_3$ *where*

$$\delta_3 = \frac{1+\alpha E_2}{1-\beta(1-\mu_2)}. \tag{3.3}$$

Proof. We first define $\mathcal{G} \in \mathcal{F}$ as the set of functions $f : \mathcal{X} \to \mathbb{R}$ which have the above properties. Note that $\mathcal{G}$ is non-empty; for example consider the function

$$f_0(x^0, x^1, x^2) = (x^0 + x^1)\frac{1+\alpha E_1}{\mu_1} + x^2\frac{1+\alpha E_2}{\mu_2}. \tag{3.4}$$

We have

$$f_0(x^0+1,x^1,x^2)-f_0(x^0,x^1,x^2)=\frac{1+\alpha E_1}{\mu_1}\geq\delta_1,$$
$$f_0(x^0,1,x^2)-f_0(x^0,0,x^2)=\frac{1+\alpha E_1}{\mu_1}\geq\delta_2,$$
$$f_0(x^0,x^1,1)-f_0(x^0,x^1,0)=\frac{1+\alpha E_2}{\mu_2}\geq\delta_3,$$

which shows that f_0 satisfies all of the properties in Lemma 1, and therefore $f_0\in\mathcal{G}$. Next consider a function $f\in\mathcal{G}$. First we will show that $\min_u f(P_u x)\in\mathcal{G}$. Note that $\min_{u_a} f(P_{u_a}x_a)-\min_{u_b} f(P_{u_b}x_b)\geq\delta$ if, for any valid control decision P_i under x_a, there exists a valid decision P_j under x_b such that $f(P_i x_a)-f(P_j x_b)\geq\delta$, since $f(P_j x_b)\geq\min_{u_b} f(P_{u_b}x_b)$ and therefore $f(P_i x_a)-\min_{u_b} f(P_{u_b}x_b)\geq\delta$ as well.

1. Define $x_a=(x^0+1,x^1,x^2), x_b=(x^0,x^1,x^2)$. We now explore every possible action on x_a and show that for each of them a valid action on x_b exists such that the difference $\geq\delta_1$.

   For $P_{u_a}=P_h$, we have

   $$f(P_h x_a)-f(P_h x_b)=f(x^0+1,x^1,x^2)-f(x^0,x^1,x^2)\geq\delta_1.$$

   For $P_{u_a}=P_1$, we have

   $$f(P_1 x_a)-f(P_h x_b)=f(x^0,1,x^2)-f(x^0,0,x^2)\geq\delta_2\geq\delta_1.$$

   For $P_{u_a}=P_2$, we have

   $$f(P_2 x_a)-f(P_h x_b)=f(x^0,x^1,1)-f(x^0,x^1,0)\geq\delta_3\geq\delta_1.$$

   For $P_{u_a}=P_b$, noting that $x_a\in Dom(P_b)$ implies $x_b\in Dom(P_1)$ since $x^0\geq 1$ and $x^1=0$, we have

   $$f(P_b x_a)-f(P_1 x_b)=f(x^0-1,1,1)-f(x^0-1,1,0)\geq\delta_3\geq\delta_1.$$

2. Define $x_a = (x^0, 1, x^2)$ and $x_b = (x^0, 0, x^2)$. We explore all possible actions on x_a noting that P_1 and P_b are not valid decisions under x_a.

   For $P_{u_a} = P_h$, we have

   $$f(P_h x_a) - f(P_h x_b) = f(x^0, 1, x^2) - f(x^0, 0, x^2) \geq \delta_2.$$

   For $P_{u_a} = P_2$, we have

   $$f(P_2 x_a) - f(P_2 x_b) = f(x^0 - 1, 1, 1) - f(x^0 - 1, 0, 1) \geq \delta_2.$$

3. Define $x_a = (x^0, x^1, 1)$ and $x_b = (x^0, x^1, 0)$. We explore all possible actions on x_a noting that P_2 and P_b are not valid decisions under x_a.

   For all $P_{u_a} = P_h$, we have

   $$f(P_h x_a) - f(P_h x_b) = f(x^0, x^1, 1) - f(x^0, x^1, 0) \geq \delta_3.$$

   For $P_{u_a} = P_1$, we have

   $$f(P_1 x_a) - f(P_1 x_b) = f(x^0 - 1, 1, 1) - f(x^0 - 1, 1, 0) \geq \delta_3.$$

The above shows that $\min_u f(P_u x) \in \mathcal{G}$ for any function $f \in \mathcal{G}$. Next we proceed with value iteration to show that $Tf \in \mathcal{G}$.

1. We start by showing that Property 1 in the Lemma is invariant under T. After

transformation, $f(x_a)$ and $f(x_b)$ can be written as:

$$\begin{aligned} Tf(x^0+1,x^1,x^2) &= x^0+1+x^1+x^2+\alpha E_1 x^1+\alpha E_2 x^2 \\ &\quad +\beta\lambda \min_{u_0} f(P_{u_0}A(x^0+1,x^1,x^2)) \\ &\quad +\beta\mu_1 \min_{u_1} f(P_{u_1}D_1(x^0+1,x^1,x^2)) \\ &\quad +\beta\mu_2 \min_{u_2} f(P_{u_2}D_2(x^0+1,x^1,x^2)) \\ &= x^0+1+x^1+x^2+\alpha E_1 x^1+\alpha E_2 x^2 \\ &\quad +\beta\lambda \min_{u_0} f(P_{u_0}(x^0+2,x^1,x^2)) \\ &\quad +\beta\mu_1 \min_{u_1} f(P_{u_1}(x^0+1,0,x^2)) \\ &\quad +\beta\mu_2 \min_{u_2} f(P_{u_2}(x^0+1,x^1,0)), \end{aligned} \tag{3.5}$$

$$\begin{aligned} Tf(x^0,x^1,x^2) &= x^0+x^1+x^2+\alpha E_1 x^1+\alpha E_2 x^2+\beta\lambda \min_{u_0} f(P_{u_0}A(x^0,x^1,x^2)) \\ &\quad +\beta\mu_1 \min_{u_1} f(P_{u_1}D_1(x^0,x^1,x^2))+\beta\mu_2 \min_{u_2} f(P_{u_2}D_2(x^0,x^1,x^2)) \\ &= x^0+x^1+x^2+\alpha E_1 x^1+\alpha E_2 x^2+\beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,x^1,x^2)) \\ &\quad +\beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,x^2))+\beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,x^1,0)). \end{aligned} \tag{3.6}$$

Taking the difference between Equation (3.5) and Equation (3.6),

$$\begin{aligned}
&Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) \\
&\quad = 1 + \beta\lambda(\min_{u_0} f(P_{u_0}(x^0+2,x^1,x^2)) - \min_{u_0} f(P_{u_0}(x^0+1,x^1,x^2))) \\
&\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0+1,0,x^2)) - \min_{u_1} f(P_{u_1}(x^0,0,x^2))) \\
&\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0+1,x^1,0)) - \min_{u_2} f(P_{u_2}(x^0,x^1,0))) \\
&\quad \geq 1 + \beta\lambda\delta_1 + \beta\mu_1\delta_1 + \beta\mu_2\delta_1 \\
&\quad = 1 + \beta\delta_1.
\end{aligned}$$

Since δ_1 can take the value of $\frac{1}{1-\beta}$, δ_2, or δ_3, we consider each case and show that $1+\beta\delta_1 \geq \delta_1$ for any possible value of δ_1.

If $\delta_1 = \frac{1}{1-\beta}$,

$$Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) \geq 1 + \beta\frac{1}{1-\beta} = \delta_1.$$

If $\delta_1 = \delta_2$, we have

$$\begin{aligned}
Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) &\geq 1 + \beta\frac{1+\alpha E_1}{1-\beta(1-\mu_1)} \\
&= \frac{1+\beta(\mu_1+\alpha E_1)}{1-\beta(1-\mu_1)}.
\end{aligned}$$

For $\beta \geq \frac{\alpha E_1}{\mu_1+\alpha E_1}$,

$$\frac{1+\beta(\mu_1+\alpha E_1)}{1-\beta(1-\mu_1)} \geq \frac{1+\alpha E_1}{1-\beta(1-\mu_1)} = \delta_2,$$

and therefore

$$Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) \geq \delta_2 = \delta_1.$$

Similarly for $\delta_1 = \delta_3$ and $\beta \geq \frac{\alpha E_2}{\mu_2+\alpha E_2}$,

$$Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) \geq \delta_3 = \delta_1.$$

We have shown that $Tf(x^0+1,x^1,x^2) - Tf(x^0,x^1,x^2) \geq \delta_1$ for all $\beta \in [\beta^*, 1)$ where $\beta^* = \max(\frac{\alpha E_1}{\mu_1+\alpha E_1}, \frac{\alpha E_2}{\mu_2+\alpha E_2})$, therefore the optimal value function $\lim_{n\to\infty} T^n f = J^\beta$ has the same property.

2. Next we show that Property 2 in the Lemma is also invariant under T. After transformation, $Tf(x_a)$ and $Tf(x_b)$ can be written as:

$$\begin{aligned} Tf(x^0,1,x^2) &= x^0 + 1 + x^2 + \alpha E_1 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0} A(x^0,1,x^2)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1} D_1(x^0,1,x^2)) + \beta\mu_2 \min_{u_2} f(P_{u_2} D_2(x^0,1,x^2)) \\ &= x^0 + 1 + x^2 + \alpha E_1 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,x^2)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,x^2)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)), \end{aligned} \tag{3.7}$$

$$\begin{aligned} Tf(x^0,0,x^2) &= x^0 + x^2 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0} A(x^0,0,x^2)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1} D_1(x^0,0,x^2)) + \beta\mu_2 \min_{u_2} f(P_{u_2} D_2(x^0,0,x^2)) \\ &= x^0 + x^2 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,0,x^2)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,x^2)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,0,0)). \end{aligned} \tag{3.8}$$

Taking the difference between Equation (3.7) and Equation (3.8),

$$\begin{aligned} &Tf(x^0,1,x^2) - Tf(x^0,0,x^2) \\ &\quad = 1 + \alpha E_1 + \beta\lambda(\min_{u_0} f(P_{u_0}(x^0+1,1,x^2)) - \min_{u_0} f(P_{u_0}(x^0+1,0,x^2))) \\ &\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0,0,x^2)) - \min_{u_1} f(P_{u_1}(x^0,0,x^2))) \\ &\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0,1,0)) - \min_{u_2} f(P_{u_2}(x^0,0,0))) \\ &\quad \geq 1 + \alpha E_1 + \beta\lambda\delta_2 + \beta\mu_2\delta_2 \\ &\quad = 1 + \alpha E_1 + \beta(1-\mu_1)\delta_2 = \delta_2. \end{aligned}$$

Hence we have proven that $Tf(x^0, 1, x^2) - Tf(x^0, 0, x^2) \geq \delta_2$ for any β, and therefore the optimal value function $\lim_{n\to\infty} T^n f = J^\beta$ has the same property.

3. Using the same technique used to prove Property 2, we can show that $Tf(x^0, x^1, 1) - Tf(x^0, x^1, 0) \geq \delta_3$ for any β, and therefore the optimal value function $\lim_{n\to\infty} T^n f = J^\beta$ has the same property.[1]

We have shown that $\mathcal{G}$ is non-empty and $Tf \in \mathcal{G}$ for any function $f \in \mathcal{G}$, therefore the optimal value function $\lim_{n\to\infty} T^n f = J^\beta \in \mathcal{G}$. □

3.2 Keep the Efficient Server Busy When Possible

In this section we show that it is optimal to make a scheduling decision which keeps the more efficient server busy whenever possible, i.e., ensure that a job is assigned to server 1 whenever $x \in Dom(P_1)$ or $x \in Dom(P_b)$, in the setting where the more efficient server is also faster.

Proposition 1. *When $\mu_1 > \mu_2$, there exists $\beta^* \in [0, 1)$ such that for all $\beta \in [\beta^*, 1)$, J^β has the following properties:*

1. $J^\beta(P_1 x) \leq J^\beta(P_2 x)$ *when* $x = (1, 0, 0)$,
2. $J^\beta(P_1 x) \leq J^\beta(P_h x)$ *for* $x \in Dom(P_1)$, *and*
3. $J^\beta(P_b x) \leq J^\beta(P_2 x)$ *for* $x \in Dom(P_b)$.

[1] See A.1 for the complete proof.

Proof. Consider an $f \in \mathcal{G}$ which has the above properties. Again such functions exist, for example f_0 in Equation (3.4)[2].

First we show that Property 1 (i.e., it is better to send the job to the more efficient server rather than the less efficient one when there is only one job in the queue) is invariant under T. The state under consideration is $x = (1, 0, 0)$; possible actions in this state are P_h, P_1, and P_2. We can eliminate the P_h option from consideration since P_1 would be a better choice in comparison, due to Property 2. The possibilities that remain are P_1 and P_2, and we will show that $Tf(P_1 x) \leq Tf(P_2 x)$ in this state. Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$\begin{aligned}
&Tf(0,1,0) - Tf(0,0,1) \\
&\quad = \alpha E_1 - \alpha E_2 + \beta\lambda(\min_{u_0} f(P_{u_0}(1,1,0)) - \min_{u_0} f(P_{u_0}(1,0,1))) \\
&\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(0,0,0)) - \min_{u_1} f(P_{u_1}(0,0,1))) \\
&\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(0,1,0)) - \min_{u_2} f(P_{u_2}(0,0,0))) \\
&\quad \leq \alpha E_1 - \alpha E_2 + \beta\lambda(f(0,1,1) - f(0,1,1)) + \beta\mu_1(f(0,0,0) - f(0,0,1)) \\
&\qquad + \beta\mu_2(f(0,1,0) - f(0,0,0)) \\
&\quad = \alpha E_1 - \alpha E_2 + \beta(\mu_1 - \mu_2) f(0,0,0) + \beta\mu_2 f(0,1,0) - \beta\mu_1 f(0,0,1) \\
&\quad = \alpha E_1 - \alpha E_2 + \beta(\mu_1 - \mu_2)(f(0,0,0) - f(0,1,0)) \\
&\quad \leq \alpha E_1 - \alpha E_2 + \beta(\mu_1 - \mu_2)(-\delta_2) \\
&\quad = \alpha E_1 - \alpha E_2 - \beta(\mu_1 - \mu_2)\frac{1 + \alpha E_1}{1 - \beta(1 - \mu_1)}.
\end{aligned} \tag{3.9}$$

We make use of the fact that $f(0,1,0) \leq f(0,0,1)$ to upper bound the $-\beta\mu_1 f(0,0,1)$ term with $-\beta\mu_1 f(0,1,0)$ in order to factor out $\beta(\mu_1 - \mu_2)$ between steps 3 and 4.

[2]For proof see Appendix A.3.

In order to show that $Tf(0,1,0) \leq Tf(0,0,1)$, it is sufficient to show the upper bound

$$\alpha E_1 - \alpha E_2 - \beta(\mu_1 - \mu_2)\frac{1+\alpha E_1}{1-\beta(1-\mu_1)} \leq 0,$$

which after algebraic manipulation yields the condition

$$\beta(\alpha E_1 - \alpha E_2 + \mu_1 - \mu_2 + \mu_1\alpha E_2 - \mu_2\alpha E_1) \geq \alpha E_1 - \alpha E_2.$$

For convenience, we rewrite this in the form of

$$\beta(A+B) \geq A \tag{3.10}$$

where $A = \alpha E_1 - \alpha E_2$ and $B = \mu_1 - \mu_2 + \mu_1\alpha E_2 - \mu_2\alpha E_1$. Note that B is always positive since we can derive

$$\mu_1 - \mu_2 + \mu_1\alpha E_2 - \mu_2\alpha E_1 > 0$$

from Equation (2.2). Several cases arise depending on the parameters of the system:

If $E_1 > E_2$, both sides of Equation (3.10) are positive, and we can rearrange it as

$$\beta \geq \frac{A}{A+B}.$$

Since $A < A + B$ in this case, the right hand side is a value $\in [0,1)$ and Equation (3.10) holds for all $\beta \in [\frac{A}{A+B}, 1)$.

If $E_1 \leq E_2$ and $\beta(A+B) \geq 0$, Equation (3.10) holds for all $\beta \in [0,1)$. On the other hand if $\beta(A+B) < 0$, rearranging Equation (3.10) yields the condition

$$\beta \leq \frac{A}{A+B}.$$

In this case $\frac{A}{A+B} > 1$, so again the condition holds for all $\beta \in [0,1)$. Combining all cases, we can define

$$\beta^* = \begin{cases} \frac{\alpha E_1 - \alpha E_2}{\alpha E_1 - \alpha E_2 + \mu_1 - \mu_2 + \mu_1\alpha E_2 - \mu_2\alpha E_1}, & \text{if } E_1 > E_2 \\ 0, & \text{otherwise,} \end{cases}$$

which gives us $Tf(0,1,0) \leq Tf(0,0,1)$, and by extension $J^\beta(0,1,0) - J^\beta(0,0,1) \leq 0$, for all $\beta \in [\beta^*, 1)$, as required. Next we show that Property 2, i.e., sending a job to the more efficient server is better than holding it in the queue, is also invariant under T. The setting under consideration is $x \in Dom(P_1)$, i.e., $x = (x^0, 0, x^2)$, where $x^0 \geq 1$. After applying the transformation to $f(P_h x)$ and $f(P_1 x)$, we get

$$\begin{aligned}
Tf(P_1x) &= x^0 + x^2 + \alpha E_1 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0} A P_1 x) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1} D_1 P_1 x) + \beta\mu_2 \min_{u_2} f(P_{u_2} D_2 P_1 x), \\
Tf(P_hx) &= x^0 + x^2 + \alpha E_2 x^2 + \beta\lambda \min_{u_0} f(P_{u_0} A P_h x) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1} D_1 P_h x) + \beta\mu_2 \min_{u_2} f(P_{u_2} D_2 P_h x).
\end{aligned}$$

Taking the difference between them, we get

$$Tf(P_1x) - Tf(P_hx) = \alpha E_1 + \beta\lambda(\min_{u_0} f(P_{u_0} A P_1 x) - \min_{u_0} f(P_{u_0} A P_h x)) \tag{3.11}$$

$$+ \beta\mu_1(\min_{u_1} f(P_{u_1} D_1 P_1 x) - \min_{u_1} f(P_{u_1} D_1 P_h x)) \tag{3.12}$$

$$+ \beta\mu_2(\min_{u_2} f(P_{u_2} D_2 P_1 x) - \min_{u_2} f(P_{u_2} D_2 P_h x)). \tag{3.13}$$

For the λ term, we have

$$\begin{aligned}
&\beta\lambda(\min_{u_0} f(P_{u_0} A P_1 x) - \min_{u_0} f(P_{u_0} A P_h x)) \\
&\quad = \beta\lambda(\min_{u_0} f(P_{u_0}(x^0, 1, x^2)) - \min_{u_0} f(P_{u_0}(x^0 + 1, 0, x^2))).
\end{aligned} \tag{3.14}$$

When $x^2 = 0$, the right hand side of Equation (3.14) becomes

$$\beta\lambda(\min_{u_0} f(P_{u_0}(x^0, 1, 0)) - \min_{u_0} f(P_{u_0}(x^0 + 1, 0, 0))).$$

While any $P \in \{P_h, P_1, P_2, P_b\}$ would be a valid action for the subtrahend, the action which incurs the minimum cost can only be either P_1 or P_b; P_h and P_2 are eliminated from

consideration since $f(P_1 x) \leq f(P_h x)$, and $f(P_b x) \leq f(P_2 x)$ by Property 3. In the former case, the subtrahend evaluates to $f(x^0, 1, 0)$, and we have

$$\min_{u_0} f(P_{u_0}(x^0, 1, 0)) \leq f(x^0, 1, 0);$$

in the latter case the subtrahend evaluates to $f(x^0 - 1, 1, 1)$, and we have

$$\min_{u_0} f(P_{u_0}(x^0, 1, 0)) \leq f(x^0 - 1, 1, 1).$$

In either case, the minuend evaluates to a quantity no larger than the value of the subtrahend, and thus the right hand side of Equation (3.14) is non-positive.

When $x^2 = 1$, Equation (3.14) can be written as

$$\beta\lambda(f(x^0, 1, 1) - \min_{u_0} f(P_{u_0}(x^0 + 1, 0, 1))).$$

The state in the subtrahend, $(x^0 + 1, 0, 1)$, is in $Dom(P_1)$ and $Dom(P_h)$; however we know that $f(P_1 x) \leq f(P_h x)$, thus the action which incurs the minimum cost in this state must be P_1. The subtrahend therefore evaluates to $f(x^0, 1, 1)$, resulting in a difference of 0 in Equation (3.14).

In every case,

$$\min_{u_0} f(P_{u_0}(x^0, 1, x^2)) \leq \min_{u_0} f(P_{u_0}(x^0 + 1, 0, x^2)),$$

and the entire λ term ≤ 0.

For the μ_2 term, we have

$$\begin{aligned} &\beta\mu_2(\min_{u_2} f(P_{u_2} D_2 P_1 x) - \min_{u_2} f(P_{u_2} D_2 P_h x)) \\ &\quad = \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0 - 1, 1, 0)) - \min_{u_2} f(P_{u_2}(x^0, 0, 0))). \end{aligned} \tag{3.15}$$

When $x^0 = 1$, this can be rewritten as

$$\beta\mu_2(f(0,1,0) - \min_{u_2} f(P_{u_2}(1,0,0))).$$

Since we have $f(0,1,0) \leq f(0,0,1)$ and $f(P_1 x) \leq f(P_h x)$, the action which incurs the minimum cost in the subtrahend must be P_1; the subtrahend therefore evaluates to $f(0,1,0)$ as well, leaving a difference of 0.

When $x^0 \geq 2$, we can define $\bar{x^0} = x^0 - 1$ and rewrite the above as

$$\beta\mu_2(\min_{u_2} f(P_{u_2}(\bar{x^0},1,0)) - \min_{u_2} f(P_{u_2}(\bar{x^0}+1,0,0))),$$

to which the same arguments used for the λ term can be applied to show that the entire μ_2 term ≤ 0.

Lastly for the μ_1 term, we have

$$\begin{aligned}&\beta\mu_1(\min_{u_1} f(P_{u_1} D_1 P_1 x) - \min_{u_1} f(P_{u_1} D_1 P_h x)) \\ &\quad = \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0 - 1, 0, x^2)) - \min_{u_1} f(P_{u_1}(x^0, 0, x^2))),\end{aligned}$$

and from the proof of Lemma 1 we have

$$\min_{u_1} f(P_{u_1}(x^0+1, x^1, x^2)) - \min_{u_1} f(P_{u_1}(x^0, x^1, x^2)) \geq \delta_1.$$

This produces an upper bound of $\beta\mu_1(-\delta_1)$ for the μ_1 term.

Combining the results of the λ, μ_1, and μ_2 terms, we can define the lower bound

$$Tf(P_h x) - Tf(P_1 x) \geq -\alpha E_1 + \beta\mu_1\delta_1$$

for Equation (3.11). Recall from Lemma 1 that $\delta_1 = \min(\frac{1}{1-\beta}, \delta_2, \delta_3)$. To facilitate a proof for the lower bound $-\alpha E_1 + \beta\mu_1\delta_1 \geq 0$, we would like to choose β^* such that $\delta_1 = \delta_2$, i.e., $\delta_2 \leq \frac{1}{1-\beta}$ and $\delta_2 \leq \delta_3$. After algebraic manipulation, the condition for $\delta_2 \leq \frac{1}{1-\beta}$ is

$$\beta \geq \frac{\alpha E_1}{\alpha E_1 + \mu_1},$$

and the condition for $\delta_2 \leq \delta_3$ is identical to Equation (3.10), thus the previous analytical results on the values of β which satisfy this inequality can be applied here as well. Combining the bounds on β for both conditions, we can define

$$\beta^* = \begin{cases} \max\left(\frac{\alpha E_1}{\alpha E_1+\mu_1}, \frac{\alpha E_1-\alpha E_2}{\alpha E_1-\alpha E_2+\mu_1-\mu_2+\mu_1\alpha E_2-\mu_2\alpha E_1}\right), & \text{if } E_1 > E_2 \\ \frac{\alpha E_1}{\alpha E_1+\mu_1}, & \text{otherwise,} \end{cases}$$

which gives us $\delta_1 = \delta_2$ for $\beta \in [\beta^*, 1)$. The lower bound on $Tf(P_h x) - Tf(P_1 x)$ becomes

$$\begin{aligned} Tf(P_1x) - Tf(P_hx) &\leq \alpha E_1 - \beta^*\mu_1\delta_2 \\ &= \alpha E_1 - \beta^*\mu_1\frac{1+\alpha E_1}{1-\beta^*(1-\mu_1)} \\ &= \frac{\alpha E_1(1-\beta^*+\beta^*\mu_1) - \beta^*\mu_1(1+\alpha E_1)}{1-\beta^*(1-\mu_1)} \\ &= \frac{\alpha E_1 - \beta^*(\alpha E_1+\mu_1)}{1-\beta^*(1-\mu_1)} \\ &\leq \frac{\alpha E_1 - \frac{\alpha E_1}{\alpha E_1+\mu_1}(\alpha E_1+\mu_1)}{1-\beta^*(1-\mu_1)} \\ &\leq 0, \end{aligned}$$

which implies $J^\beta(P_1x) - J^\beta(P_hx) \leq 0$ for $\beta \in [\beta^*, 1)$ as required.

Finally we will show that Property 3, i.e., it is better to assign jobs to both servers rather than only the less efficient server when there are at least two jobs in the queue, is invariant under T as well. For this property the setting is $x \in Dom(P_b)$, i.e. $x = (x^0, 0, 0), x^0 \geq 2$. We begin by defining $x^* = (x^0 - 1, 0, 1)$. The value functions under comparison are $f(x^0 - 2, 1, 1)$ and $f(x^0 - 1, 0, 1)$; after applying the transformation, we can rewrite the states using x^* as

$$Tf(P_bx) = Tf(x^0 - 2, 1, 1) = Tf(P_1(x^0 - 1, 0, 1)) = Tf(P_1x^*)$$

and

$$Tf(P_2x) = Tf(x^0 - 1, 0, 1) = Tf(P_h(x^0 - 1, 0, 1)) = Tf(P_hx^*).$$

We have shown above that $Tf(P_1x) \leq Tf(P_hx)$ for any $x \in Dom(P_1)$ and $\beta \geq \beta^*$. Noting that $x \in Dom(P_b) \implies x^* \in Dom(P_1)$, we can write $Tf(P_1x^*) \leq Tf(P_hx^*)$ or equivalently $Tf(P_bx) \leq Tf(P_2x)$, which implies that $J^\beta(P_bx) \leq J^\beta(P_2x)$ for $\beta \in [\beta^*, 1)$ as required. □

We have shown that when there is only one job in the queue, it is always better to assign it to the more efficient server when possible, as $J^\beta(0, 1, 0) \leq J^\beta(0, 0, 1)$ and $J^\beta(P_1(x)) \leq J^\beta(P_h(x))$. In the general case when there is more than one job in the queue, we have shown that $J^\beta(P_b(x)) \leq J^\beta(P_2(x))$ in addition to $J^\beta(P_1(x)) \leq J^\beta(P_h(x))$. A proof for $J^\beta(P_1(x)) \leq J^\beta(P_2(x))$ for the general case is not necessary, as the inequality is not required for subsequent proofs, and P_1 and P_b are the only viable actions both of which keep the more efficient server active whenever possible.

3.3 Send Jobs to the Less Efficient Server When Queue Length Exceeds Threshold

To prove that the optimal policy has the second characteristic of a threshold type policy, i.e., jobs should be schedule for the less efficient server once the queue length exceeds some threshold, it is sufficient to show [2] that when $x \in Dom(P_2)$, the cost benefit of withholding a job from the less efficient server over sending it is monotonically decreasing with queue length. In other words, when $x^0 = 0$,

$$J^\beta(0, 0, 1) - J^\beta(0, 1, 0) \geq J^\beta(0, 1, 1) - J^\beta(1, 1, 0), \tag{3.16}$$

and when $x^0 \geq 1$,

$$J^\beta(x^0 - 1, 1, 1) - J^\beta(x^0, 1, 0) \geq J^\beta(x^0, 1, 1) - J^\beta(x^0 + 1, 1, 0). \tag{3.17}$$

When there is only one job in the queue prior to the control decision and server 1 is idle, it is optimal to assign the job to server 1 by Proposition 1. When server 1 is occupied, the cost benefit of keeping server 2 idle is $J^\beta(x^0 - 1, 1, 1) - J^\beta(x^0, 1, 0)$; if this benefit is positive, it is better not to assign jobs to server 2. With increasing x^0 the cost benefit monotonically decreases, and if at $x^0 = x_a^0$ there is no longer a positive cost benefit in keeping server 2 idle, i.e., $J^\beta(x_a^0 - 1, 1, 1) - J^\beta(x_a^0, 1, 0) \leq 0$, then the threshold has been exceeded and it becomes optimal to assign jobs to both servers for all $x^0 \geq x_a^0$. We now show that J^β has the properties put forth in Proposition 2, number 1 and 2 of which are equivalent to Equations (3.17) and (3.16) respectively. The following proofs follow along the same lines as Koole's proof in [2], with necessary modifications to account for the costs of using each server.

Proposition 2. *There exists $\beta^* \in [0, 1)$ such that for all $\beta \in [\beta^*, 1)$, the optimal value function J^β has the following properties:*

1. $J^\beta(x^0, 1, 0) + J^\beta(x^0, 1, 1) \leq J^\beta(x^0 + 1, 1, 0) + J^\beta(x^0 - 1, 1, 1)$ *for* $x^0 \geq 1$
2. $J^\beta(0, 1, 0) + J^\beta(0, 1, 1) \leq J^\beta(1, 1, 0) + J^\beta(0, 0, 1)$
3. $J^\beta(x^0, 1, 0) + J^\beta(x^0 - 1, 1, 1) \leq J^\beta(x^0 - 1, 1, 0) + J^\beta(x^0, 1, 1)$ *for* $x^0 \geq 1$
4. $J^\beta(0, 1, 0) + J^\beta(0, 0, 1) \leq J^\beta(0, 0, 0) + J^\beta(0, 1, 1)$.

Proof. We take the set of functions $\mathcal{G}$ as defined in Section 3 and once more constrain it to functions with the above properties, noting that $\mathcal{G}$ is non-empty[3].

[3]For proof see Appendix A.4.

Now, consider an $f \in \mathcal{G}$. For convenience, we first derive three additional properties of f. Summing properties 1 and 3 yields

5. $2f(x^0, 1, 0) \leq f(x^0 + 1, 1, 0) + f(x^0 - 1, 1, 0)$,

summing properties 1 and 3 with x^0 replaced by $x^0 + 1$ in 3 yields

6. $2f(x^0, 1, 1) \leq f(x^0 + 1, 1, 1) + f(x^0 - 1, 1, 1)$,

and summing properties 2 and 4 yields

7. $2f(0, 1, 0) \leq f(1, 1, 0) + f(0, 0, 0)$.

We begin by showing that $\min_u f(P_u(x)) \in \mathcal{G}$. As an aside, note that in the derivations in this chapter we make use of the properties from Proposition 1 that $f(0, 1, 0) \leq f(0, 0, 1)$ and $f(x^0, 1, x^2) \leq f(x^0 + 1, 0, x^2)$ to replace terms where the minimum cost action is taken from a state where $x \in Dom(P_1)$, with the equivalent term where the minimum cost action is taken after first assigning a job from the queue to server 1. In other words,

$$\min_u f(P_u(1, 0, 0)) = f(0, 1, 0),$$

and

$$\min_u f(P_u(x^0, 0, x^2)) = \min_u f(P_u(x^0 - 1, 1, x^2))$$

when $x^0 \geq 1$.

1. If $\min_u f(P_u(x^0 + 1, 1, 0)) = f(x^0 + 1, 1, 0)$, we have

$$\begin{aligned}
&\min_u f(P_u(x^0, 1, 0)) + \min_u f(P_u(x^0, 1, 1)) \\
&\quad \leq f(x^0, 1, 0) + f(x^0, 1, 1) \\
&\quad \leq f(x^0 + 1, 1, 0) + f(x^0 - 1, 1, 1) \\
&\quad = \min_u f(P_u(x^0 + 1, 1, 0)) + \min_u f(P_u(x^0 - 1, 1, 1)).
\end{aligned}$$

On the other hand if $\min_u f(P_u(x^0+1,1,0)) = f(x^0,1,1)$, we have

$$\begin{aligned}
&\min_u f(P_u(x^0,1,0)) + \min_u f(P_u(x^0,1,1)) \\
&\quad \le f(x^0-1,1,1) + f(x^0,1,1) \\
&\quad = \min_u f(P_u(x^0+1,1,0)) + \min_u f(P_u(x^0-1,1,1)).
\end{aligned}$$

In both cases,

$$\begin{aligned}
&\min_u f(P_u(x^0,1,0)) + \min_u f(P_u(x^0,1,1)) \\
&\quad \le \min_u f(P_u(x^0+1,1,0)) + \min_u f(P_u(x^0-1,1,1)).
\end{aligned}$$

2. If $\min_u f(P_u(1,1,0)) = f(1,1,0)$, we have

$$\begin{aligned}
\min_u f(P_u(0,1,0)) + \min_u f(P_u(0,1,1)) &= f(0,1,0) + f(0,1,1) \\
&\le f(1,1,0) + f(0,0,1) \\
&= \min_u f(P_u(1,1,0)) + \min_u f(P_u(0,0,1))
\end{aligned}$$

and if $\min_u f(P_u(1,1,0)) = f(0,1,1)$, we have

$$\begin{aligned}
\min_u f(P_u(0,1,0)) + \min_u f(P_u(0,1,1)) &= f(0,1,0) + f(0,1,1) \\
&\le f(0,1,1) + f(0,0,1) \\
&= \min_u f(P_u(1,1,0)) + \min_u f(P_u(0,0,1)),
\end{aligned}$$

making use of $f(0,1,0) \le f(0,0,1)$ to upper bound the right hand side in step 1. In both cases,

$$\min_u f(P_u(0,1,0)) + \min_u f(P_u(0,1,1)) \le \min_u f(P_u(1,1,0)) + \min_u f(P_u(0,0,1)).$$

3. If $\min_u f(P_u(x^0 - 1, 1, 0)) = f(x^0 - 1, 1, 0)$, we have

$$\begin{aligned}
&\min_u f(P_u(x^0, 1, 0)) + \min_u f(P_u(x^0 - 1, 1, 1)) \\
&\quad \leq f(x^0, 1, 0) + f(x^0 - 1, 1, 1) \\
&\quad \leq f(x^0 - 1, 1, 0) + f(x^0, 1, 1) \\
&\quad = \min_u f(P_u(x^0 - 1, 1, 0)) + \min_u f(P_u(x^0, 1, 1))
\end{aligned}$$

and if $\min_u f(P_u(x^0 - 1, 1, 0)) = f(x^0 - 2, 1, 1)$, we have

$$\begin{aligned}
&\min_u f(P_u(x^0, 1, 0)) + \min_u f(P_u(x^0 - 1, 1, 1)) \\
&\quad \leq f(x^0 - 1, 1, 1) + f(x^0 - 1, 1, 1) \\
&\quad \leq f(x^0 - 2, 1, 1) + f(x^0, 1, 1) \\
&\quad = \min_u f(P_u(x^0 - 1, 1, 0)) + \min_u f(P_u(x^0, 1, 1)).
\end{aligned}$$

In both cases,

$$\begin{aligned}
&\min_u f(P_u(x^0, 1, 0)) + \min_u f(P_u(x^0 - 1, 1, 1)) \\
&\quad \leq \min_u f(P_u(x^0 - 1, 1, 0)) + \min_u f(P_u(x^0, 1, 1)).
\end{aligned}$$

4. This property is straightforward to show, since no actions are possible for any of the states under consideration:

$$\begin{aligned}
\min_u f(P_u(0, 1, 0)) + \min_u f(P_u(0, 0, 1)) &= f(0, 1, 0) + f(0, 0, 1) \\
&\leq f(0, 0, 0) + f(0, 1, 1) \\
&= \min_u f(P_u(0, 0, 0)) + \min_u f(P_u(0, 1, 1)).
\end{aligned}$$

Finally we proceed with value iteration and show that $Tf \in \mathcal{G}$.

1. We start by showing that the Property 1 in the Proposition is invariant under T.

$$Tf(x^0,1,0) = x^0 + 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,0)) \\ + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)),$$

$$Tf(x^0,1,1) = x^0 + 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,1)) \\ + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)),$$

$$Tf(x^0+1,1,0) = x^0 + 2 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+2,1,0)) \\ + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0+1,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0+1,1,0)),$$

$$Tf(x^0-1,1,1) = x^0 + 1 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0,1,1)) \\ + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-1,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0-1,1,0)).$$

Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$\begin{aligned} &Tf(x^0,1,0) + Tf(x^0,1,1) - Tf(x^0+1,1,0) - Tf(x^0-1,1,1) \\ &\quad = \beta\lambda(\min_{u_0} f(P_{u_0}(x^0+1,1,0)) + \min_{u_0} f(P_{u_0}(x^0+1,1,1)) \\ &\qquad - \min_{u_0} f(P_{u_0}(x^0+2,1,0)) - \min_{u_0} f(P_{u_0}(x^0,1,1))) \\ &\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0,0,0)) + \min_{u_1} f(P_{u_1}(x^0,0,1)) \\ &\qquad - \min_{u_1} f(P_{u_1}(x^0+1,0,0)) - \min_{u_1} f(P_{u_1}(x^0-1,0,1))) \\ &\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0,1,0)) + \min_{u_2} f(P_{u_2}(x^0,1,0)) \\ &\qquad - \min_{u_2} f(P_{u_2}(x^0+1,1,0)) - \min_{u_2} f(P_{u_2}(x^0-1,1,0))). \end{aligned}$$

The λ and μ_2 terms can be shown to be ≤ 0 using Properties 1 and 5 respectively. For the expression within the μ_1 term, we have two cases depending on x^0:

if $x^0 = 1$, we have

$$\begin{aligned}
&\min_{u_1} f(P_{u_1}(1,0,0)) + \min_{u_1} f(P_{u_1}(1,0,1)) \\
&\qquad\qquad - \min_{u_1} f(P_{u_1}(2,0,0)) - \min_{u_1} f(P_{u_1}(0,0,1)) \\
&= \min_{u_1} f(P_{u_1}(0,1,0)) + \min_{u_1} f(P_{u_1}(0,1,1)) \\
&\qquad\qquad - \min_{u_1} f(P_{u_1}(1,1,0)) - \min_{u_1} f(P_{u_1}(0,0,1)) \\
&\leq 0 \qquad \text{by Property 2.}
\end{aligned}$$

If $x^0 \geq 2$, we have

$$\begin{aligned}
&\min_{u_1} f(P_{u_1}(x^0,0,0)) + \min_{u_1} f(P_{u_1}(x^0,0,1)) \\
&\qquad\qquad - \min_{u_1} f(P_{u_1}(x^0+1,0,0)) - \min_{u_1} f(P_{u_1}(x^0-1,0,1)) \\
&= \min_{u_1} f(P_{u_1}(x^0-1,1,0)) + \min_{u_1} f(P_{u_1}(x^0-1,1,1)) \\
&\qquad\qquad - \min_{u_1} f(P_{u_1}(x^0,1,0)) - \min_{u_1} f(P_{u_1}(x^0-2,1,1)) \\
&\leq 0 \qquad \text{by Property 1.}
\end{aligned}$$

In either case above, the right hand side ≤ 0 for all $x^0 \geq 1$ and $\beta \in [\beta^*, 1)$[4], therefore

$$Tf(x^0,1,0) + Tf(x^0,1,1) \leq Tf(x^0+1,1,0) + Tf(x^0-1,1,1)$$

as required.

[4] β^* being the value of β above which properties from Lemma 1 and Proposition 1 hold.

2. Next we show that the Property 2 in the Proposition is invariant under T.

$$\begin{aligned}
Tf(0,1,0) &= 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)) \\
&= 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,0)) \\
&\quad + \beta\mu_1 f(0,0,0) + \beta\mu_2 f(0,1,0),
\end{aligned}$$

$$\begin{aligned}
Tf(0,1,1) &= 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)) \\
&= 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,1)) \\
&\quad + \beta\mu_1 f(0,0,1) + \beta\mu_2 f(0,1,0),
\end{aligned}$$

$$\begin{aligned}
Tf(1,1,0) &= 2 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(1,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)) \\
&= 2 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,0)) \\
&\quad + \beta\mu_1 f(0,1,0) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)),
\end{aligned}$$

$$\begin{aligned}
Tf(0,0,1) &= 1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,0,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,0,0)) \\
&= 1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(0,1,1)) \\
&\quad + \beta\mu_1 f(0,0,1) + \beta\mu_2 f(0,0,0).
\end{aligned}$$

Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$\begin{aligned}
&Tf(0,1,0) + Tf(0,1,1) - Tf(1,1,0) - Tf(0,0,1) \\
&\quad = \alpha E_1 + \beta\lambda(\min_{u_0} f(P_{u_0}(1,1,0)) + \min_{u_0} f(P_{u_0}(1,1,1)) \\
&\qquad\qquad - \min_{u_0} f(P_{u_0}(2,1,0)) - \min_{u_0} f(P_{u_0}(0,1,1))) \\
&\qquad + \beta\mu_1(f(0,0,0) + f(0,0,1) - f(0,1,0) - f(0,0,1)) \\
&\qquad + \beta\mu_2(f(0,1,0) + f(0,1,0) - \min_{u_2} f(P_{u_2}(1,1,0)) - f(0,0,0)) \\
&\quad \leq \alpha E_1 - \beta\mu_1\delta_2 + \beta\mu_2(2f(0,1,0) - \min_{u_2} f(P_{u_2}(1,1,0)) - f(0,0,0)).
\end{aligned}$$

For the μ_2 term, if $\min_{u_2} f(P_{u_2}(1,1,0)) = f(1,1,0)$, we have

$$\begin{aligned}
2f(0,1,0)) &\leq f(1,1,0) + f(0,0,0) \\
&= \min_{u_2} f(P_{u_2}(1,1,0)) + f(0,0,0)
\end{aligned}$$

and if $\min_{u_2} f(P_{u_2}(1,1,0)) = f(0,1,1)$, we have

$$\begin{aligned}
2f(0,1,0)) &\leq f(0,1,0) + f(0,0,1) \\
&\leq f(0,0,0) + f(0,1,1) \\
&= \min_{u_2} f(P_{u_2}(1,1,0)) + f(0,0,0).
\end{aligned}$$

In either case,

$$2f(0,1,0)) \leq \min_{u_2} f(P_{u_2}(1,1,0)) + f(0,0,0).$$

This leaves us with the upper bound

$$\begin{aligned}
&Tf(0,1,0) + Tf(0,1,1) - Tf(1,1,0) - Tf(0,0,1) \\
&\quad \leq \alpha E_1 - \beta\mu_1\delta_2 \\
&\quad = \alpha E_1 - \beta\mu_1 \frac{1 + \alpha E_1}{1 - \beta(1 - \mu_1)}
\end{aligned}$$

For $\beta \geq \frac{\alpha E_1}{\alpha E_1 + \mu_1}$,

$$\alpha E_1 - \beta \mu_1 \frac{1 + \alpha E_1}{1 - \beta(1 - \mu_1)} \leq 0.$$

This implies

$$Tf(0,1,0) + Tf(0,1,1) \leq Tf(1,1,0) + Tf(0,0,1)$$

for $\beta \in [\beta^*, 1)$, where β^* is the maximum of $\frac{\alpha E_1}{\alpha E_1 + \mu_1}$ and the value above which Lemma 1 and Proposition 1 hold, as required.

3. Next we show that the Property 3 in the Proposition is invariant under T. We have two cases depending on the value of x^0. If $x^0 \geq 2$,

$$\begin{aligned}
Tf(x^0,1,0) &= x^0 + 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)) \\
&= x^0 + 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-1,1,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)),
\end{aligned}$$

$$\begin{aligned}
Tf(x^0-1,1,1) &= x^0 + 1 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-1,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0-1,1,0)) \\
&= x^0 + 1 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-2,1,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0-1,1,0)),
\end{aligned}$$

$$\begin{aligned}
Tf(x^0-1,1,0) &= x^0 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-1,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0-1,1,0)) \\
&= x^0 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-2,1,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0-1,1,0)),
\end{aligned}$$

$$
\begin{aligned}
Tf(x^0,1,1) &= x^0 + 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)) \\
&= x^0 + 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(x^0+1,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(x^0-1,1,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(x^0,1,0)).
\end{aligned}
$$

Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$
\begin{aligned}
&Tf(x^0,1,0) + Tf(x^0-1,1,1) - Tf(x^0-1,1,0) - Tf(x^0,1,1) \\
&\quad = \beta\lambda(\min_{u_0} f(P_{u_0}(x^0+1,1,0)) + \min_{u_0} f(P_{u_0}(x^0,1,1)) \\
&\qquad - \min_{u_0} f(P_{u_0}(x^0,1,0)) - \min_{u_0} f(P_{u_0}(x^0+1,1,1))) \\
&\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0-1,1,0)) + \min_{u_1} f(P_{u_1}(x^0-2,1,1)) \\
&\qquad - \min_{u_1} f(P_{u_1}(x^0-2,1,0)) - \min_{u_1} f(P_{u_1}(x^0-1,1,1))) \\
&\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0,1,0)) + \min_{u_2} f(P_{u_2}(x^0-1,1,0)) \\
&\qquad - \min_{u_2} f(P_{u_2}(x^0-1,1,0)) - \min_{u_2} f(P_{u_2}(x^0,1,0))).
\end{aligned}
$$

The right hand side ≤ 0 since the expressions within each of the terms ≤ 0. However a special case emerges when $x^0 = 1$ due to the $(x^0-2,1,x^2)$ states, and the transformations must be re-evaluated as follows:

$$
\begin{aligned}
Tf(1,1,0) &= 2 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(1,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)) \\
&= 2 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,1,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)),
\end{aligned}
$$

$$Tf(0,1,1) = 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,1)) + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)),$$

$$Tf(0,1,0) = 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,0)) + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)),$$

$$\begin{aligned} Tf(1,1,1) &= 3 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,1)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(1,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)) \\ &= 3 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(2,1,1)) \\ &\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,1,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(1,1,0)). \end{aligned}$$

Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$\begin{aligned} &Tf(1,1,0) + Tf(0,1,1) - Tf(0,1,0) - Tf(1,1,1) \\ &\quad = \beta\lambda(\min_{u_0} f(P_{u_0}(2,1,0)) + \min_{u_0} f(P_{u_0}(1,1,1)) \\ &\qquad - \min_{u_0} f(P_{u_0}(1,1,0)) - \min_{u_0} f(P_{u_0}(2,1,1))) \\ &\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(0,1,0)) + \min_{u_1} f(P_{u_1}(0,0,1)) \\ &\qquad - \min_{u_1} f(P_{u_1}(0,0,0)) - \min_{u_1} f(P_{u_1}(0,1,1))) \\ &\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(1,1,0)) + \min_{u_2} f(P_{u_2}(0,1,0)) \\ &\qquad - \min_{u_2} f(P_{u_2}(0,1,0)) - \min_{u_2} f(P_{u_2}(1,1,0))) \end{aligned}$$

$$
\begin{aligned}
&\leq \beta\lambda(\min_{u_0} f(P_{u_0}(2,1,0)) + \min_{u_0} f(P_{u_0}(1,1,1)) \\
&\qquad - \min_{u_0} f(P_{u_0}(1,1,0)) - \min_{u_0} f(P_{u_0}(2,1,1))) \\
&\quad + \beta\mu_1(f(0,1,0) + f(0,0,1) - f(0,0,0) - f(0,1,1)) \\
&\quad + \beta\mu_2(\min_{u_2} f(P_{u_2}(1,1,0)) + \min_{u_2} f(P_{u_2}(0,1,0)) \\
&\qquad - \min_{u_2} f(P_{u_2}(0,1,0)) - \min_{u_2} f(P_{u_2}(1,1,0))).
\end{aligned}
$$

Again all the terms ≤ 0, therefore

$$
Tf(x^0,1,0) + Tf(x^0-1,1,1) \leq Tf(x^0-1,1,0) + Tf(x^0,1,1)
$$

holds for any $x^0 \geq 1$ as required.

4. Finally we show that Property 4 in the Proposition is invariant under T.

$$
\begin{aligned}
Tf(0,1,0) &= 1 + \alpha E_1 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,0)) \\
&\qquad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)),
\end{aligned}
$$

$$
\begin{aligned}
Tf(0,0,1) &= 1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,0,1)) \\
&\qquad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,0,0)) \\
&= 1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(0,1,1)) \\
&\qquad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,0,0)),
\end{aligned}
$$

$$
\begin{aligned}
Tf(0,0,0) &= \beta\lambda \min_{u_0} f(P_{u_0}(1,0,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,0,0)) \\
&= \beta\lambda \min_{u_0} f(P_{u_0}(0,1,0)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,0)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,0,0)),
\end{aligned}
$$

$$
\begin{aligned}
Tf(0,1,1) &= 2 + \alpha E_1 + \alpha E_2 + \beta\lambda \min_{u_0} f(P_{u_0}(1,1,1)) \\
&\quad + \beta\mu_1 \min_{u_1} f(P_{u_1}(0,0,1)) + \beta\mu_2 \min_{u_2} f(P_{u_2}(0,1,0)).
\end{aligned}
$$

Taking the difference between the terms corresponding to the left hand side of the inequality and the right, we get

$$
\begin{aligned}
&Tf(0,1,0) + Tf(0,0,1) - Tf(0,0,0) - Tf(0,1,1) \\
&\quad = \beta\lambda(\min_{u_0} f(P_{u_0}(1,1,0)) + \min_{u_0} f(P_{u_0}(0,1,1)) \\
&\qquad - \min_{u_0} f(P_{u_0}(0,1,0)) - \min_{u_0} f(P_{u_0}(1,1,1))).
\end{aligned}
$$

The expression inside the λ term ≤ 0 by Property 3, and so the right hand side of the inequality ≤ 0 for $\beta \in [\beta^*, 1)$[5]. Therefore

$$
Tf(0,1,0) + Tf(0,0,1) \leq Tf(0,0,0) + Tf(0,1,1)
$$

as required.

□

We have shown that $\mathcal{G}$ is non-empty and that $Tf \in \mathcal{G}$ for any function $f \in \mathcal{G}$. This implies that the optimal value function $\lim_{n\to\infty} T^n f = J^\beta \in \mathcal{G}$ and therefore satisfies the

[5]See footnote 4.

properties in Proposition 2, including Property 1 and 2 which are equivalent to Equation (3.17) and (3.16) respectively. This shows that jobs should be sent to the less efficient server once the queue length exceeds some threshold, and this result in conjunction with the conclusion from Section 3.2 that the more efficient server should be kept busy whenever possible, we can conclude that the optimal policy is of threshold type in the setting where $\mu_1 > \mu_2$. This result generalizes the findings of Koole [2], i.e., the optimal policy is of threshold type when the goal is to minimize only the response time of a job through the system.

Chapter 4

Extension to $\mu_1 < \mu_2$ with Small λ

In this chapter we extend our results from Chapter 3 to the setting where $\mu_1 < \mu_2$, i.e., the more efficient server is slower, with the additional restriction of small λ. In Section 4.1 we will show algebraically that $J^\beta(0,1,0) \leq J^\beta(0,0,1)$ in this setting. In Section 4.2 we discuss the work required to extend the results from Chapter 3 to this setting. We were unable to obtain an analytical proof under large λ, or a value iteration proof of the inequality.

4.1 $J^\beta(0,1,0) \leq J^\beta(0,0,1)$

In this section we will show that when there is only one job in the queue, it is better to send it to server 1 rather than 2.

Lemma 2. *When $\mu_1 < \mu_2$, for sufficiently small λ and fixed $\beta \in [0,1)$, $J^\beta(P_1 x) \leq J^\beta(P_2 x)$ when $x = (1,0,0)$.*

Proof. Rather than using value iteration, we will algebraically show that Lemma 2 is a property of the optimal cost function J^β in this setting. We begin by noting that $J^\beta(x)$ is increasing with λ for all $x \in \mathcal{X}$[1]. We take the difference between the Bellman equations for each resultant state:

$$\begin{aligned}
&J^\beta(0,1,0) - J^\beta(0,0,1) \\
&\quad = \alpha E_1 - \alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,1,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1))) \\
&\qquad + \beta\mu_1(J^\beta(0,0,0) - J^\beta(0,0,1)) + \beta\mu_2(J^\beta(0,1,0) - J^\beta(0,0,0)) \\
&\quad = \alpha E_1 - \alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,1,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1))) \\
&\qquad + \beta(\mu_1 - \mu_2)(J^\beta(0,0,0) - J^\beta(0,0,1)) \\
&\qquad + \beta\mu_2(J^\beta(0,1,0) - J^\beta(0,0,1)).
\end{aligned}$$

Rearranging, we get

$$\begin{aligned}
&J^\beta(0,1,0) - J^\beta(0,0,1) \\
&\quad = \frac{1}{1-\beta\mu_2}\Big(\alpha E_1 - \alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,1,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1))) \\
&\qquad\qquad + \beta(\mu_1 - \mu_2)(J^\beta(0,0,0) - J^\beta(0,0,1))\Big). \quad (4.1)
\end{aligned}$$

Note that the difference $J^\beta(0,0,0) - J^\beta(0,0,1)$ in the $\beta(\mu_1 - \mu_2)$ term can also be written as the difference between Bellman equations as follows:

$$\begin{aligned}
J^\beta(0,0,0) - J^\beta(0,0,1) = -\alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,0,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1))) \\
+ \beta\mu_1(J^\beta(0,0,0) - J^\beta(0,0,1)).
\end{aligned}$$

[1]For proof see Appendix A.2.

Rearranging, we get

$$J^\beta(0,0,0) - J^\beta(0,0,1) = \frac{-\alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,0,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1)))}{1-\beta\mu_1}. \quad (4.2)$$

Substituting Equation (4.2) into (4.1) yields

$$\begin{aligned}
&J^\beta(0,1,0) - J^\beta(0,0,1) \\
&\quad = \frac{1}{(1-\beta\mu_1)(1-\beta\mu_2)}\Bigg((1-\beta\mu_1)(\alpha E_1 - \alpha E_2) \\
&\qquad + (1-\beta\mu_1)\beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,1,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1))) \\
&\qquad + \beta(\mu_1-\mu_2)\Big(-\alpha E_2 + \beta\lambda(\min_{u_0} J^\beta(P_{u_0}(1,0,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1)))\Big)\Bigg) \\
&\quad = \frac{1}{(1-\beta\mu_1)(1-\beta\mu_2)}\Bigg(\alpha E_1(1-\beta\mu_1) - \alpha E_2(1-\beta\mu_2) \\
&\qquad + \beta\lambda\Bigg(\min_{u_0} J^\beta(P_{u_0}(1,1,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1)) \\
&\qquad\qquad + \beta\mu_1(\min_{u_0} J^\beta(P_{u_0}(1,0,0)) - \min_{u_0} J^\beta(P_{u_0}(1,1,0))) \\
&\qquad\qquad - \beta\mu_2(\min_{u_0} J^\beta(P_{u_0}(1,0,0)) - \min_{u_0} J^\beta(P_{u_0}(1,0,1)))\Bigg)\Bigg).
\end{aligned}$$

Since we have $\alpha E_1(1-\beta\mu_1) - \alpha E_2(1-\beta\mu_2) \leq 0 \; \forall \beta \in [0,1)$ and the $\beta\lambda$ terms $\to 0$ as $\lambda \to 0$ since $J^\beta(x)$ is increasing in λ, for sufficiently small λ and fixed β, $J^\beta(0,1,0) \leq J^\beta(0,0,1)$. □

4.2 Future Work to Show Optimality of Threshold Policy When $\mu_1 < \mu_2$

In this section we discuss the additional result that would have to be shown in order to prove optimality of a threshold policy for the setting where $\mu_1 < \mu_2$.

The value iteration proof for Property 1 of Proposition 1 requires $\mu_1 > \mu_2$, since we cannot upper bound $Tf(0,1,0) - Tf(0,0,1)$ with $\alpha E_1 - \alpha E_2 + \beta(\mu_1 - \mu_2)(-\delta_2)$ as in Equation 3.9 if $\beta(\mu_1 - \mu_2)$ is a negative coefficient. In its place we have Lemma 2, which shows the equivalent result when $\mu_1 < \mu_2$ and λ is small. The rest of the proofs for Propositions 1 and 2 will hold under $\mu_1 < \mu_2$ if one can show via value iteration that $J^\beta(0,1,0) \leq J^\beta(0,0,1)$; the current proof of this in Lemma 2 uses a direct algebraic approach rather than value iteration.

Chapter 5

Simulation Results

5.1 Simulation Setup

In order to validate our system model as well as to evaluate the practical applications of a power-aware threshold policy, we created a discrete event simulator in C++[1] to realize our discrete state model from Chapter 2. The simulator consists of a system of two heterogeneous servers as described in Chapter 1, with system parameters

$$\mu_1 = 10,\ \mu_2 = 100,\ E_1 = 5,\ E_2 = 100,\ \bar{\alpha} = 100,$$

chosen based on response time and power usage measurements taken from real computers while satisfying the requirement that server 1 more energy efficient than server 2, and $\bar{\alpha}$ is chosen such that the inequality $\frac{1+\alpha E_1}{\mu_1} \leq \frac{1+\alpha E_2}{\mu_2}$, where $\alpha = \lambda\bar{\alpha}$, holds for all $\lambda \in [1, \mu_1+\mu_2)$. We also chose a setting where $\mu_1 < \mu_2$ in order to validate our results from Chapter 4.

We implemented two scheduling policies for our system: the naive first-come-first-serve

[1] See Appendix B for full source code listing.

policy, which assigns a job from the queue whenever a server is idle, beginning with server 1; and the threshold policy, which assigns a job to server 1 if possible, and server 2 only if the queue length has exceeded the threshold parameter m. For each policy, we sweep across a range of arrival rates from 1 up to but not including $\mu_1 + \mu_2$, and serve a total of 10 million jobs at each arrival rate. In order to allow the system to stabilize, we first wait for 30% of the jobs to be completed before collecting any data.

Workload is simulated by scheduling exponentially distributed arrival events at rate λ; whenever a job arrives, the scheduler puts it in a common queue, then makes a scheduling decision depending on the policy in effect. Once a job starts being served, the server is set to a busy state and the job's departure is scheduled exponentially using the server's service rate μ. When a departure occurs, the server's state is set to idle, data on the job's service and response times are recorded if the system is stable, and the scheduler again checks if either server should begin service on a job.

The main performance metrics we collected during simulation include the mean response time of a job through the system $\bar{T}$ and mean power consumption in the system $\bar{E}$. These metrics are used in calculating the cost J of each policy using our cost function

$$J = \lambda \bar{T} + \lambda \bar{\alpha} \bar{E}, \tag{5.1}$$

where $\bar{T}$ is the mean response time and $\bar{\alpha}\bar{E}$ is the weighted power usage.

We ran simulation experiments using each scheduling policy from un-unified rates $\lambda = 1$ through $\mu_1 + \mu_2 - 1$; for the threshold policy, we adjust the threshold parameter m from 0 to 10 at each rate, completing a full simulation run of 10 million jobs for each parameter value. For each arrival rate and parameter value we record the mean response time and power usage, and compute the cost J with those parameters using the cost function above.

In the remainder of this chapter we will present the results of the simulations and discuss our findings.

5.2 Threshold Policy Compared

In this section we will compare the performance of the threshold policy against that of the naive policy. For the each arrival rate under the threshold policy, we calculate the cost J using Equation (5.1) for each threshold parameter from 0 to 10, and choose the parameter which incurs the lowest cost as the optimal threshold for that arrival rate. The costs of the threshold policy with optimal thresholds and first-come-first-serve policy as well as the percentage difference between them are shown in Figure 5.1.

From the results we can see that the threshold policy's improvement over the naive policy peaks at 29.19% when $\lambda = 8$, and drops off as λ increases. This matches our expectations since as workload increases, keeping one server idle becomes less beneficial as jobs start building up in the queue and incur longer waiting times. At very high λ, both servers need to be kept busy to keep up with the high arrival rate and clear jobs from the system, reducing the optimal threshold and essentially matching the naive policy which also keeps both servers busy whenever possible. While we were only able to provide preliminary analytical results limited to low λ for the optimality of the threshold policy in the setting where $\mu_1 < \mu_2$, these empirical results show that the threshold policy is strictly better than the naive policy for the two-server system modelled here.

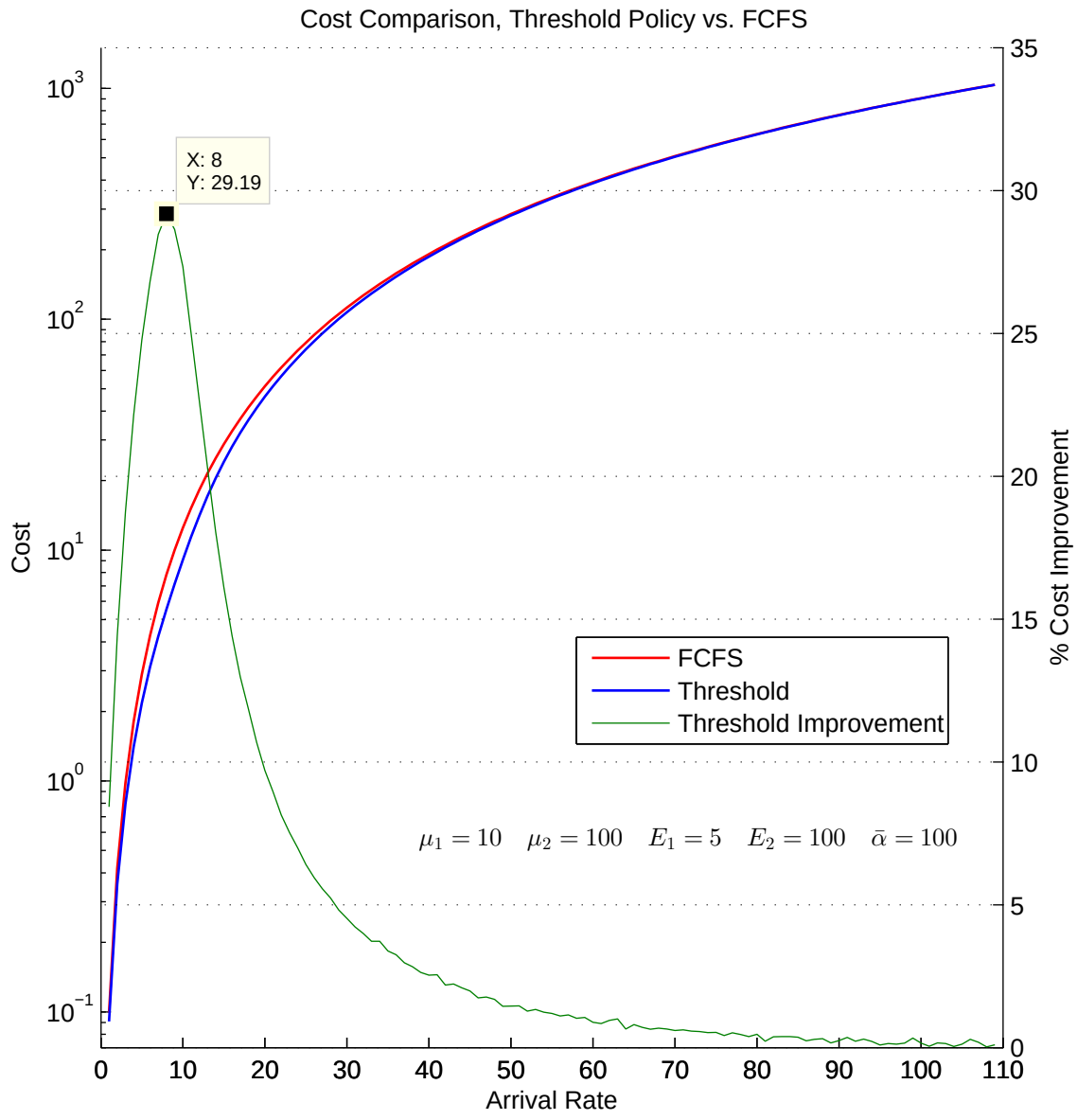

Figure 5.1: Threshold policy with optimal thresholds compared against first-come-first-serve.

Chapter 6

Conclusion

In this thesis we presented a method of balancing highly variable workloads using a system of heterogeneous servers, with the goal of exploiting its capability to reduce power consumption during times of low workloads while maintaining responsive performance during peak loads. Extending from previous work [1][2] on the optimality of the threshold policy, we used analytical methods to show that it is also optimal in such a system when the additional consideration of power is added in the setting where the more efficient server is also faster, and obtained preliminary results for a restricted setting where the slower server is more efficient and arrival rate is very low.

The simulated comparison between a threshold policy and a naive first-come-first-serve policy shows that the threshold policy incurs an appreciable decrease in costs over the naive policy when the arrival rate is low, while matching its performance when the arrival rate is high, meeting our goal of increasing efficiency during low workloads. Opportunities remain for future work to be done on this topic, including a value iteration proof of $J^{\beta}(P_1(x)) \leq J^{\beta}(P_2(x))$ in the $\mu_1 < \mu_2$ setting and a method of calculating the optimal threshold given

system parameters

APPENDICES

Appendix A

Detailed Proofs

A.1 Proof for Property 3 of Lemma 1

$$\begin{aligned} Tf(x^0,x^1,1) &= x^0+x^1+1+\alpha E_1 x^1+\alpha E_2+\beta\lambda\min_{u_0} f(P_{u_0}A(x^0,x^1,1)) \\ &\quad +\beta\mu_1\min_{u_1} f(P_{u_1}D_1(x^0,x^1,1))+\beta\mu_2\min_{u_2} f(P_{u_2}D_2(x^0,x^1,1)) \\ &= x^0+x^1+1+\alpha E_1 x^1+\alpha E_2+\beta\lambda\min_{u_0} f(P_{u_0}(x^0+1,x^1,1)) \\ &\quad +\beta\mu_1\min_{u_1} f(P_{u_1}(x^0,0,1))+\beta\mu_2\min_{u_2} f(P_{u_2}(x^0,x^1,0)), \end{aligned} \tag{A.1}$$

$$\begin{aligned} Tf(x^0,x^1,0) &= x^0+x^1+\alpha E_1 x^1+\beta\lambda\min_{u_0} f(P_{u_0}A(x^0,x^1,0)) \\ &\quad +\beta\mu_1\min_{u_1} f(P_{u_1}D_1(x^0,x^1,0))+\beta\mu_2\min_{u_2} f(P_{u_2}D_2(x^0,x^1,0)) \\ &= x^0+x^1+\alpha E_1 x^1+\beta\lambda\min_{u_0} f(P_{u_0}(x^0+1,x^1,0)) \\ &\quad +\beta\mu_1\min_{u_1} f(P_{u_1}(x^0,0,0))+\beta\mu_2\min_{u_2} f(P_{u_2}(x^0,x^1,0)). \end{aligned} \tag{A.2}$$

Taking the difference between Equation A.1 and Equation A.2,

$$\begin{aligned}
&Tf(x^0,x^1,1) - Tf(x^0,x^1,0) \\
&\quad = 1 + \alpha E_2 + \beta\lambda(\min_{u_0} f(P_{u_0}(x^0+1,x^1,1)) - \min_{u_0} f(P_{u_0}(x^0+1,x^1,0))) \\
&\qquad + \beta\mu_1(\min_{u_1} f(P_{u_1}(x^0,0,1)) - \min_{u_1} f(P_{u_1}(x^0,0,0))) \\
&\qquad + \beta\mu_2(\min_{u_2} f(P_{u_2}(x^0,x^1,0)) - \min_{u_2} f(P_{u_2}(x^0,x^1,0))) \\
&\quad \geq 1 + \alpha E_2 + \beta\lambda\delta_3 + \beta\mu_1\delta_3 \\
&\quad = 1 + \alpha E_2 + \beta(1-\mu_2)\delta_3 = \delta_3.
\end{aligned}$$

Hence $Tf(x^0,x^1,1) - Tf(x^0,x^1,0) \geq \delta_3$ for any β, and therefore the optimal value function $\lim_{n\to\infty} T^n f = J^\beta$ has the same property.

A.2 $J^\beta(x)$ is increasing with λ

Proof. Consider a system with service rates μ_1 and μ_2; we wish to compare two configurations of this system, one under arrival rate λ_a, labelled configuration a, and the other under arrival rate λ_b, labelled configuration b, where $\lambda_a < \lambda_b$. The service rates $(\mu_{1i}, \mu_{2i}), i \in \{a, b\}$ in each configuration are normalized such that $\lambda_i + \mu_{1i} + \mu_{2i} = 1$. Let $J_i^\beta(x)$ be the optimal cost starting from state x in configuration i. Let π be the optimal policy for configuration b. The Bellman equation for configuration b in state x is

$$J_b^\beta(x) = c(x) + \beta\lambda_b J_b^\beta(PAx) + \beta\mu_{1b} J_b^\beta(PD_1x) + \beta\mu_{2b} J_b^\beta(PD_2x)$$

where P represents the optimal control decision in states Ax, D_1x, and D_2x chosen by π. Take $J_b^\beta(x)$ for all $x \in \mathcal{X}$ and stack them into an ∞-dimensional vector J_b^β and stack all

stage costs into another ∞-dimensional vector c. The Bellman equation for all states can be written as

$$J_b^\beta = c + \beta \bar{P}_b J_b^\beta \tag{A.3}$$

where $\bar{P}_b$ is a ∞-dimensional matrix, each row having λ_b, μ_{1b}, and μ_{2b} entries corresponding to the optimal destination states under π.

Now consider the same policy π in configuration a. The Bellman equation for the system in state x is

$$J_{a,\pi}^\beta(x) = c(x) + \beta\lambda_a J_{a,\pi}^\beta(PAx) + \beta\mu_{1a} J_{a,\pi}^\beta(PD_1x) + \beta\mu_{2a} J_{a,\pi}^\beta(PD_2x)$$

where $J_{a,\pi}^\beta(x)$ denotes the cost of the policy π from state x, and P denotes the action chosen by π from Ax, D_1x, and D_2x. This can be written as

$$J_{a,\pi}^\beta = c + \beta \bar{P}_{a,\pi} J_{a,\pi}^\beta \tag{A.4}$$

where $\bar{P}_{a,\pi}$ is simply $\bar{P}_b$ with λ_a, μ_{1a}, and μ_{2a} instead of λ_b, μ_{1b}, and μ_{2b}. Equation A.3 and A.4 can be written as

$$(I - \beta \bar{P}_{a,\pi}) J_{a,\pi}^\beta = c \tag{A.5}$$

and

$$(I - \beta \bar{P}_b) J_b^\beta = c. \tag{A.6}$$

Next, define

$$\Delta = J_b^\beta - J_{a,\pi}^\beta.$$

Equating Equation A.5 and A.6, we get

$$\begin{aligned}
(I - \beta \bar{P}_{a,\pi}) J_{a,\pi}^\beta &= (I - \beta \bar{P}_b) J_b^\beta \\
&= (I - \beta \bar{P}_b) J_{a,\pi}^\beta + (I - \beta \bar{P}_b)\Delta \\
\beta(\bar{P}_b - \bar{P}_{a,\pi}) J_{a,\pi}^\beta &= (I - \beta \bar{P}_b)\Delta.
\end{aligned} \tag{A.7}$$

Note that each row of $\bar{P}_b - \bar{P}_{a,\pi}$ contains the terms $\lambda_b - \lambda_a$, $\mu_{1b} - \mu_{1a}$, and $\mu_{2b} - \mu_{2a}$ as the non-zero elements, and the entire row sums to 0. Since $\lambda_a < \lambda_b$, $\lambda_b - \lambda_a$ is positive and the other two terms are negative, since the ratio of μ_{1i} to μ_{2i} are the same for $i \in \{a, b\}$ therefore both μ_{1a} and μ_{2a} must both increase to maintain $\lambda_a + \mu_{1a} + \mu_{2a} = 1$. For each state x^i, let the destination states after PAx^i, PD_1x^i, and PD_2x^i be x^i_λ, x^i_1, and x^i_2 respectively. Note that $J^\beta_{a,\pi}(x^i_\lambda) \geq \max(J^\beta_{a,\pi}(x^i_1), J^\beta_{a,\pi}(x^i_2))$ since the state after an arrival has at least one more job in the system than the states after a departure.

Since the elements in each row of $\bar{P}_b - \bar{P}_{a,\pi}$ sum to zero and the positive term $\lambda_b - \lambda_a$ is multiplied by $J^\beta_{a,\pi}(x^i_\lambda)$ which is a larger multiplier than the ones applied to the negative terms, we have

$$\begin{aligned} &\beta(\bar{P}_b - \bar{P}_{a,\pi})J^\beta_{a,\pi} \geq 0 \text{ (elementwise)} \\ \implies &(I - \beta\bar{P}_b)\Delta \geq 0 \text{ (elementwise)}, \end{aligned}$$

which implies there exists a vector $d \geq 0$ such that

$$\Delta = (I - \beta\bar{P}_b)^{-1}d.$$

Since $||\beta\bar{P}_b|| < 1$, we have

$$\begin{aligned} &(I - \beta\bar{P}_b)^{-1} = I + \beta\bar{P}_b + \beta^2{\bar{P}_b}^2 + \cdots \geq 0 \\ \implies &\Delta = J^\beta_b - J^\beta_{a,\pi} \geq 0 \text{ (elementwise)} \end{aligned}$$

and therefore

$$J^\beta_b \geq J^\beta_{a,\pi} \geq J^\beta_a$$

where J^β_a denotes the vector of costs $J^\beta_a(x)$ in configuration a under the optimal policy for all states $x \in \mathcal{X}$, therefore $J^\beta_a(x) < J^\beta_b(x)$ for any x. This shows that the optimal cost $J^\beta(x)$ is increasing with λ for all $x \in \mathcal{X}$, as required. □

A.3 The set of functions with properties listed in Proposition 1 is non-empty

Proof. Using f_0 as defined in Equation 3.4, when $x = (1, 0, 0)$:

$$f_0(P_1 x) - f_0(P_2 x) = \frac{1 + \alpha E_1}{\mu_1} - \frac{1 + \alpha E_2}{\mu_2} < 0.$$

When $x \in Dom(P_1)$:

$$f_0(P_1 x) - f_0(P_h x) = -\frac{1 + \alpha E_1}{\mu_1} + \frac{1 + \alpha E_1}{\mu_1} = 0.$$

When $x \in Dom(P_b)$,

$$f_0(P_b x) - f_0(P_2 x) = (\frac{1 + \alpha E_1}{\mu_1} + \frac{1 + \alpha E_2}{\mu_2}) - (\frac{1 + \alpha E_1}{\mu_1} + \frac{1 + \alpha E_2}{\mu_2}) = 0.$$

□

A.4 The set of functions with properties listed in Proposition 2 is non-empty

Proof. Using f_0 as defined in Equation 3.4:

$$\begin{aligned}
&f_0(x^0, 1, 0) + f_0(x^0, 1, 1) - f_0(x^0 + 1, 1, 0) - f_0(x^0 - 1, 1, 1) \\
&= (x^0 + 1)\frac{1 + \alpha E_1}{\mu_1} + (x^0 + 1)\frac{1 + \alpha E_1}{\mu_1} + \frac{1 + \alpha E_2}{\mu_2} \\
&\quad - (x^0 + 2)\frac{1 + \alpha E_1}{\mu_1} - x^0\frac{1 + \alpha E_1}{\mu_1} - \frac{1 + \alpha E_2}{\mu_2} \\
&= 0,
\end{aligned}$$

$$
\begin{aligned}
&f_0(0,1,0) + f_0(0,1,1) - f_0(1,1,0) - f_0(0,0,1) \\
&\quad = \frac{1+\alpha E_1}{\mu_1} + \frac{1+\alpha E_1}{\mu_1} + \frac{1+\alpha E_2}{\mu_2} - 2\frac{1+\alpha E_1}{\mu_1} - \frac{1+\alpha E_2}{\mu_2} \\
&\quad = 0,
\end{aligned}
$$

$$
\begin{aligned}
&f_0(x^0,1,0) + f_0(x^0-1,1,1) - f_0(x^0-1,1,0) - f_0(x^0,1,1) \\
&\quad = (x^0+1)\frac{1+\alpha E_1}{\mu_1} + x^0\frac{1+\alpha E_1}{\mu_1} + \frac{1+\alpha E_2}{\mu_2} \\
&\qquad - x^0\frac{1+\alpha E_1}{\mu_1} - (x^0+1)\frac{1+\alpha E_1}{\mu_1} + \frac{1+\alpha E_2}{\mu_2} \\
&\quad = 0,
\end{aligned}
$$

$$
\begin{aligned}
&f_0(0,1,0) + f_0(0,0,1) - f_0(0,0,0) - f_0(0,1,1) \\
&\quad = \frac{1+\alpha E_1}{\mu_1} + \frac{1+\alpha E_2}{\mu_2} - \frac{1+\alpha E_1}{\mu_1} - \frac{1+\alpha E_2}{\mu_2} \\
&\quad = 0.
\end{aligned}
$$

□

Appendix B

Simulator Source Code

options.h

```
#pragma once

// Default number of requests to make per run
#define TOTAL_REQUESTS 10000000
#define TOTAL_RUN_TIME 100000
#define WARMUP_FACTOR 0.3

// Slow Server
#define S1_SERVICE_RATE 10
// Fast Server
#define S2_SERVICE_RATE 100

// Power parameters
#define S1_IDLE_POWER 0
#define S1_MAX_POWER 5
#define S2_IDLE_POWER 0
```

```
#define S2_MAX_POWER 100

// Timeout parameters
#define S1_TIMEOUT 0
#define S2_TIMEOUT 0
```

functions.h

```
#pragma once
#include <cstdlib>
#include <cmath>
#include <vector>
using namespace std;

double calc_exp(double rate);
double calc_mean(vector<double> v);
```

functions.cpp

```
#include "functions.h"
using namespace std;

// Calculate exponentially distributed next event time based on rate
double calc_exp(double rate) {
    // Generate a random double between 0 and 1, not including 0
    double u;
    do { u = (double) rand() / RAND_MAX; }
    while (u == 0);
    return -log(u) / rate;
}

// Calculate mean for array
```

```
14 double calc_mean(vector<double> v) {
       double sum = 0;
16     for (int i = 0; i < v.size(); i++) {
           sum = sum + v[i];
18     }
       return sum / (double) v.size();
20 }
```

main.h

```
   #pragma once
2  #include "options.h"
   #ifndef __GNUC__
4  #include "getopt.h"
   #else
6  #include <getopt.h>
   #endif
8  #include <algorithm>
   #include <queue>
10 #include <vector>
   #include <stdio.h>
12 #include <cstdlib>
   #include <time.h>
14 #include <string>
   #include <list>
16 #include <iostream>
   #include <fstream>
18 #include <map>
   #include <cmath>
20 #include <sstream>  //for std::istringstream
   #include <iterator> //for std::istream_iterator
```

```
22 #include <vector>   //for std::vector

24 using namespace std;

26 enum EVENT_TYPE {
       ARRIVAL,
28     START_SERVICE,
       DEPARTURE
30 };

32 enum STATUS {
       IDLE = 0,
34     BUSY = 1
   };
36
   class sim_job {
38 public:
       double arrival_t;
40     sim_job(double t) : arrival_t(t) {}

42 };

44 class server {
   public:
46     int n;
       STATUS status;
48     queue<sim_job> job_queue;
       int completed;
50     double serv_rate;
       vector<double> resp_times;
```

```
52      double idle_t;
        double serv_t;
54      double last_service_start;
        double last_service_stop;
56
        server(int n, double s) : n(n), serv_rate(s) {
58          status = IDLE;
            completed = 0;
60          idle_t = 0;
            serv_t = 0;
62          last_service_start = 0;
            last_service_stop = 0;
64      }

66      ~server() {
            resp_times.clear();
68          while(!job_queue.empty()) {
                job_queue.pop();
70          }
        }
72  };

74  class sim_event {
    public:
76      double event_time;
        double wait_t;
78      double serv_t;
        EVENT_TYPE event_type;
80      server * s;
```

```
82      bool operator < (sim_event other) const {
            return event_time < other.event_time;
84      }

86      // Constructors
        sim_event() {}
88      sim_event(EVENT_TYPE p, double t) : event_type(p), event_time(t) {}
};
90
class Compare {
92  public:
        bool operator() (const sim_event lhs, const sim_event rhs) {
94          return lhs.event_time > rhs.event_time;
        }
96  };

98  void init();
    void initsim();
100 void arrival_routine();
    void start_service();
102 void start_service_routine(server * s);
    void departure_routine(sim_event * e);
104
    /* System parameters */
106 // Total requests to make per run
    int total_requests;
108 // Request rate range
    double low_rate, high_rate, rate_step;
110 // Scheduling policy
    string scheduler;
```

```
112 // Parameter for some policies
    double param;
114 // Threshold
    double thresh_low, thresh_high;
116 double timeout;
    // Output file
118 char * outfile_path;
    ofstream outfile;
120 string param_fname;
    /* ----------------------*/
122
    /* Statistical counters */
124 int total_complete;
    vector<double> all_resp;
126 /* ----------------------*/

128 // Servers
    server * s1;
130 server * s2;
    // Global queues
132 priority_queue<sim_event, vector<sim_event>, Compare> event_set;
    queue<sim_job> common_job_queue;
134 // Main sim clock
    double sim_clock;
136 // Current status
    // Keep track of the server the previous job was scheduled for
138 server * last_server;
    double curr_rate;
140 bool warmup;
    double warmup_time;
```

main.cpp

```
#include "main.h"
#include "functions.h"
using namespace std;

/*
Usage: sim <options>
    Options:
        -l      lowest request rate
        -h      highest request rate
        -t      request rate step
        -r      number of requests to make per run, default TOTAL_REQUESTS
        -s      scheduling policy, valid options are
            "FCFS"  first come first serve
            "TH"    threshold method
            "S1"    only use server 1
            "S2"    only use server 2
            "QL"  assign jobs depending on each server's queue length
      "RR"  round robin
        -p      parameter for scheduling policies
            QL    queue length difference between server 1 and 2
            TH      queue length must be longer than param to start serving
    jobs to server 2
        -o      timeout value, number of seconds a job will wait in the
    queue
                before being discarded with a response time of 0
        -f      output file
*/
int main(int argc, char *argv[]) {
    // Initialization
```

```
    init();

    // Parse options
    extern char *optarg;
    extern int optind, optopt;
    int option_char;

    // Invokes member function `int operator ()(void);'
    while ((option_char = getopt (argc, argv, "l:h:t:r:s:p:o:f:x:y:")) !=
-1) {
        switch (option_char) {
            case 'l': low_rate = atof(optarg); break;
            case 'h': high_rate = atof(optarg); break;
            case 't': rate_step = atof (optarg); break;
            case 'r': total_requests = atoi (optarg); break;
            case 's': scheduler = optarg; break;
            case 'p': param = atof(optarg); break;
            case 'o': timeout = atof(optarg); break;
            case 'f': outfile_path = optarg; break;
            case 'x': thresh_low = atof (optarg); break;
            case 'y': thresh_high = atof (optarg); break;
            case '?': fprintf (stderr, "usage: %s <options>\n", argv[0]);
        }
    }

    // Print header
    ostringstream os;
    os << "rate\tavg_resp1\tavg_resp2\tavg_resp\tutil_s1\tutil_s2\
tavg_power_s1\tavg_power_s2\tavg_power\tthresh" << endl;
    cout << os.str();
```

```
    if(outfile_path != "") {
57      outfile.open (outfile_path, ios::trunc);
        if(outfile.is_open()) {
59          outfile << os.str();
            outfile.close();
61      }
    }
63

    for(curr_rate = low_rate; curr_rate <= (high_rate + 0.05); curr_rate +=
    rate_step) {
65      double i;
        for(i = thresh_low; i <= thresh_high + 0.05; i++) {
67          if (thresh_low != thresh_high) { param = i; }
            initsim();
69          // Schedule first arrival
            event_set.push(sim_event(ARRIVAL, sim_clock + calc_exp(curr_rate
    )));
71          // Main loop
            while(total_complete < total_requests) {
73              // Remove the next scheduled event from the event set
                sim_event e = event_set.top();
75              event_set.pop();
                sim_clock = e.event_time;
77

                // Execute the event routine depending on type
79              switch(e.event_type) {
                case ARRIVAL:
81                  arrival_routine();
                    break;
83              case START_SERVICE:
```

```
                start_service_routine(e.s);
85              break;
            case DEPARTURE:
87              departure_routine(&e);
                break;
89          default:
                break;
91          }
        }
93      // Calculate stats
        double s1_avg_resp = calc_mean(s1->resp_times);
95      double s2_avg_resp = calc_mean(s2->resp_times);
        double avg_resp = calc_mean(all_resp);
97
        double s1_util = s1->serv_t / (sim_clock - warmup_time);
99      double s2_util = s2->serv_t / (sim_clock - warmup_time);
        double s1_idle = (sim_clock - warmup_time) - s1->serv_t;
101     double s2_idle = (sim_clock - warmup_time) - s2->serv_t;

103     if(scheduler == "S1") { s2_idle = 0; }
        if(scheduler == "S2") { s1_idle = 0; }
105
        double s1_idle_power = s1_idle * S1_IDLE_POWER / (sim_clock -
    warmup_time);
107     double s1_serv_power = s1_util * S1_MAX_POWER;
        double s2_idle_power = s2_idle * S2_IDLE_POWER / (sim_clock -
    warmup_time);
109     double s2_serv_power = s2_util * S2_MAX_POWER;
        double s1_power = s1_idle_power + s1_serv_power;
111     double s2_power = s2_idle_power + s2_serv_power;
```

```
113             // Print data
                ostringstream os;
115             os << curr_rate << "\t" << s1_avg_resp << "\t" << s2_avg_resp <<
    "\t" << avg_resp << "\t" << s1_util << "\t" << s2_util << "\t" <<
    s1_power << "\t" << s2_power << "\t" << s1_power + s2_power << "\t" << (
    int)param << endl;
                cout << os.str();
117             if(outfile_path != "") {
                    outfile.open (outfile_path, ios::app);
119                 if(outfile.is_open()) {
                        outfile << os.str();
121                     outfile.close();
                    }
123             }
            }
125     }
        return 0;
127 }

129 void init() {
        // Seed random number generator
131     srand(time(NULL));
        // Set parameters
133     total_requests = TOTAL_REQUESTS;

135     // Default options
        low_rate = high_rate = rate_step = 1;
137     thresh_low = thresh_high = 0;
        scheduler = "FCFS";
```

```
139     param = -1;
        timeout = 0;
141     outfile_path = "";
}
143
// Initialize a simulation run
145 void initsim() {
        sim_clock = 0;
147     warmup = false;

149     // Recreate servers
        delete s1;
151     delete s2;
        s1 = new server(1, S1_SERVICE_RATE);
153     s2 = new server(2, S2_SERVICE_RATE);

155     // Empty the event set
        while(!event_set.empty()) {
157         event_set.pop();
        }
159     all_resp.clear();
        total_complete = 0;
161   // Set the last job to server 2 by default
    last_server = s2;
163
        while(!common_job_queue.empty()) {
165         common_job_queue.pop();
        }
167 }
```

```
169 // Routine for processing arrival events
    void arrival_routine() {
171 #ifdef DEBUG
        printf("Arrival event\tTime %f\n", sim_clock);
173 #endif
        // Schedule the next arrival event
175     event_set.push(sim_event(ARRIVAL, sim_clock + calc_exp(curr_rate)));
        // Add new job to queue
177     server * srv;
        if (scheduler == "FCFS" || scheduler == "TH") { srv = NULL; }
179     else if(scheduler == "QL") {
        // Queue job for server 2 if server 1's queue length is longer by p
181     if(s1->job_queue.size() > s2->job_queue.size() + (int)param ) { srv = s2
       ; }
        else { srv = s1; }
183   }
      // Round robin
185   else if (scheduler == "RR") {
        if(last_server == s2) { last_server = s1; }
187         else { last_server = s2; }
        srv = last_server;
189   }
        else if (scheduler == "S1") { srv = s1; }
191     else if (scheduler == "S2") { srv = s2; }
        // Scheduler not defined??
193     else { exit(2); }
        if(srv == NULL) {
195         common_job_queue.push(sim_job(sim_clock));
            start_service();
197     }
```

```
    else {
199         srv->job_queue.push(sim_job(sim_clock));
            start_service_routine(srv);
201     }
}
203
// Check for queued jobs and start them
205 void start_service() {
    // If threshold is negative, check s2 first
207     if(( scheduler == "TH" || scheduler == "QL") && param < 0 ) {
            // If system is idle, start service
209         if(s2->status == IDLE) {
                start_service_routine(s2);
211         }
            if(s1->status == IDLE) {
213             start_service_routine(s1);
            }
215     }
    else {
217         // If system is idle, start service
            if(s1->status == IDLE) {
219             start_service_routine(s1);
            }
221         if(s2->status == IDLE) {
                start_service_routine(s2);
223         }
    }
225 }

227 // Routine for starting service on a job
```

```
void start_service_routine(server * s) {
#ifdef DEBUG
    printf("Service event\tServer %d\tTime %f\n", s->n, sim_clock);
#endif
    queue<sim_job> * q;
    if(scheduler == "FCFS" || scheduler == "TH") {
        q = &common_job_queue;
    }
    else {
        q = &s->job_queue;
    }
    if(q->empty() || s->status == BUSY) { return; }
    server * s_to_check;
    if(scheduler == "TH" && param < 0) {
        s_to_check = s1;
    } else {
        s_to_check = s2;
    }
    // For threshold method, process the job at the nonpreferred server only
     if queue length exceeds m
    if(s == s_to_check && scheduler == "TH" && (int)q->size() <= abs((int)
   param)) {
         // Check again in less-prefered server's service time
        sim_event dummyEvent = sim_event(START_SERVICE, sim_clock + calc_exp
   (s->serv_rate));
        dummyEvent.s = s;
        event_set.push(dummyEvent);
        return;
    }
```

```
255     // Remove the job from the front of the queue
        sim_job j = q->front();
257     q->pop();

259     double wait_time = sim_clock - j.arrival_t;
#ifdef DEBUG
261     printf("Wait Time %f\n", wait_time);
#endif
263     // If a timeout is specified, drop the job once the connection times out
        if(timeout > 0 && wait_time > timeout) {
265         total_complete++;
            start_service();
267         return;
        }
269
        // Set the server to busy and calculate the service time
271     s->status = BUSY;
        s->last_service_start = sim_clock;
273     s->idle_t += sim_clock - s->last_service_stop;
        double serv_t = calc_exp(s->serv_rate);
275 #ifdef DEBUG
        printf("Service Time %f\n", serv_t);
277 #endif
        // Schedule departure event and store the wait and service time for this
        job
279     sim_event d = sim_event(DEPARTURE, sim_clock + serv_t);
        d.s = s;
281     d.wait_t = wait_time;
        d.serv_t = serv_t;
283     event_set.push(d);
```

```
}

// Routine for processing departure events
void departure_routine(sim_event * e) {
#ifdef DEBUG
    printf("Departure event\tServer %d\tTime %f\n", e->s, sim_clock);
#endif
    if ( !warmup && total_complete > total_requests * WARMUP_FACTOR ) {
        warmup = true;
        warmup_time = sim_clock;
    }
    // Record stats if system is stable
    if (warmup) {
        // Increment the completed count for the job class
        // Add the wait time and service time to the totals for the job class
        double resp_time = e->wait_t + e->serv_t;
        e->s->completed++;
        e->s->resp_times.push_back(resp_time);
        e->s->serv_t += e->serv_t;
        all_resp.push_back(resp_time);
    }
    e->s->status = IDLE;
    e->s->last_service_stop = sim_clock;
    total_complete++;
    start_service();
}
```

References

[1] W. Lin and P. Kumar, "Optimal control of a queueing system with two heterogeneous servers," *IEEE Transactions on Automatic Control*, vol. 29, no. 8, pp. 696–703, 1984.

[2] G. Koole, "A simple proof of the optimality of a threshold policy in a two-server queueing system," *Systems & Control Letters*, vol. 26, no. 5, pp. 301–303, 1995.

[3] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 13–23, ACM, 2007.

[4] R. Brown *et al.*, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[5] T. Cao, S. M. Blackburn, T. Gao, and K. S. McKinley, "The yin and yang of power and performance for asymmetric hardware and managed software," in *ACM SIGARCH Computer Architecture News*, vol. 40, pp. 225–236, IEEE Computer Society, 2012.

[6] V. V. Rykov and D. V. Efrosinin, "On the slow server problem," *Automation and Remote Control*, vol. 70, no. 12, pp. 2013–2023, 2009.

[7] H. Li and X. Li, *Stochastic Orders in Reliability and Risk*. Springer, 2013.

[8] L. Kleinrock, *Queueing systems. Volume 1: Theory*. Wiley-Interscience, 1975.

[9] S. Lippman, "Semi-markov decision processes with unbounded rewards," *Management Sci.*, vol. 19, pp. 717–731, March 1973.

[10] P. R. Kumar and P. Varaiya, *Stochastic systems: estimation, identification and adaptive control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1986.

[11] J. Walrand, "A note on "Optimal control of a queuing system with two heterogeneous servers"," *Systems & Control Letters*, vol. 4, no. 3, pp. 131–134, 1984.