On Optimal Control for Energy-Aware Queueing Systems

Vincent J. Maccio
Department of Computing and Software
McMaster University
Hamilton, Ontario
Email: macciov@mcmaster.ca

Douglas G. Down
Department of Computing and Software
McMaster University
Hamilton, Ontario
Email downd@mcmaster.ca

***Abstract*—Over the past few years, energy provisioning in server farms and data-centres has become an active area of research. As such, many models have been proposed where an individual server has setup times and can switch between two different energy states (on and off). To make such models tractable, assumptions are usually made on the type of policies the system can implement. However, it is often not known if such assumptions allow for the model to capture the optimal policy, or if such a model will be strictly suboptimal. In this work we model such systems using Markov Decision Processes (MDPs) and derive several structural properties which (partially) describe the optimal policy. These properties reduce the set of feasible policies significantly, allowing one to describe the optimal policy by a set of thresholds which have considerable structure. In addition to the analysis, we discuss the current literature in the context of our results.**

I. Introduction

The energy costs of server farms and data-centres are currently increasing across North America and account for a significant amount of our energy consumption. While one may think that this is a necessity of progress, the truth is a lot of these systems are designed to handle peak loads, which results in a significant number of machines often idling. Furthermore, an idling server still consumes a notable amount of energy (approximately 70% of a working server) [2]. Therefore, one could switch a server to some state which consumes a lesser amount of energy, turn it off, hibernate etc. to save costs. However, there is a trade-off present since once a server is switched to such a state it is unable to work on present or arriving jobs, and the system performance suffers. This trade-off has motivated many authors to develop and analyse models of these systems [4], [5], [13], [17]–[19], [24], [25]. However, when dealing with multiple server systems with setup times, arriving at the optimal policy remains a challenge.

Queueing theory offers a large set of tools and results to model such systems. Specifically of interest are what are known as vacation models - such models have been studied for decades [1], [6], [12], [23]. In this context, a vacation can be seen as the setup or down time of the server. However, when seeking the optimal policy, energy-aware systems cannot always be embedded within already studied vacation models. Specifically, the vacation time or time before a server begins to start its setup is often assumed to be an independent random variable rather than state dependent. As such, in the past few years these models have been adapted for the specific context of server farms and data-centres.

Multiple server systems modelled as continuous time Markov chains have been studied in [7]–[9], [20], [22], [26]. Exact solutions for system metrics such as the expected response time and expected rate of energy consumption have been derived. However, all of these works considered specific policies, such as requiring a server to be turned off as soon as it becomes idle, and beginning to turn a server on as soon as there is a job waiting. Furthermore, where exact expressions were found, the assumption of server setups being interruptible was imposed. These models were extended in [10], [14], [16] to specifically look at single server systems. Due to the decreased complexity of the model, the optimal policy was found under general cost functions as well as general underlying distributions for the setup and service times.

We model a multiple server system with setup times as a Markov decision process (MDP). We further analyse this model to determine structural properties of the optimal policy which lead to a decreased set of candidate optimal policies, as well as several rules which can be used to easily determine if a given policy is suboptimal. The contributions of this paper include but are not limited to:

- A formal definition of an MDP modelling an energy-aware multiple server system for both interruptible and non interruptible setups.
- A proof that the optimal policy follows a partially ordered set of threshold values.
- For general cost functions, a proof that it is suboptimal to turn a server off if there is a job waiting in queue.
- A proof that if setups are interruptible and the cost function is linear in the expected response time and expected energy consumption then the optimal policy is a bulk setup policy.

II. Model and Notation

The system of interest has multiple servers, where the system manager has the ability to turn a server off that is currently on, or to start turning on a server which is currently off. For simplicity we often refer to powering a server down as turning it off, but in general the server is moving to some state

where it consumes energy at a lower rate. The key difference between these two server states is that when a server is off (hibernating, sleeping, etc.) it cannot process jobs, but when a server is on, it can.

A. Formal Model

Our model contains N homogeneous servers and a central queue. Each of the N servers can be in one of four energy states, *OFF, SETUP, IDLE, BUSY*. For ease of exposition we often refer to a server being *BUSY*, *IDLE*, *OFF*, or in *SETUP* as shorthand for a server being in energy state *BUSY*, *IDLE*, *OFF*, or *SETUP* respectively. A server is *BUSY* if it is on and processing a job. A server is *IDLE* if it is on but not processing a job. A server cannot begin processing a job unless it is *IDLE*. For each server, there are two control options. Firstly, an *IDLE* or *BUSY* server can be moved to energy state *OFF* (turning a server off). Secondly, an *OFF* server can be moved to energy state *SETUP* (begin turning a server on). Once a server is in setup, it will remain there for a time exponentially distributed with rate γ, at which time the server will become *IDLE*. In other words, each server has setup/turn-on times expected to last $1/\gamma$ time units, while turn-offs happen instantaneously.

Jobs arrive to the central queue following a Poisson process with rate λ and are processed on a first come first serve basis. If a job is at the front of the queue and at least one of the N servers is *IDLE*, then one of the *IDLE* servers is arbitrarily chosen to begin processing that job. Consequently, that server becomes *BUSY*. Job processing times are exponentially distributed with rate μ. Once a server finishes processing a job, it becomes *IDLE*, at this point the server can stay *IDLE*, immediately be switched to *OFF*, or start processing a new job, in turn making the server *BUSY* once again. It is assumed that $N\mu > \lambda$. Although the control of the system can always construct a policy in which the system is unstable, i.e. keep all servers turned off, as long as the condition $N\mu > \lambda$ holds, there exists a policy where the system is stable, i.e. all servers are always left on, and therefore we consider such a system stable.

From here we consider two options with respect to control of a server in the process of being in *SETUP*. One could add the control option that server setups can be interrupted, i.e. when a server is in *SETUP* it can be moved to energy state *OFF*. This is in contrast to having no control once a server begins turning on, i.e. once a server is in *SETUP* it will remain in *SETUP* until the server turns on. In this work both of these variations are analysed and it will be seen that the structure of the optimal policy is very sensitive to this choice. If the system does allow for a server to move from *SETUP* to *OFF*, we refer to such as a system as an *interruptible energy-aware system*. If this transition is not allowed, we simply refer to it as an *energy-aware system*.

With the previous description and notation in mind, we refer to an energy-aware system and interruptible energy-aware system as a four-tuple $S = (N, \lambda, \mu, \gamma)$. Furthermore, due to the assumption on the underlying distributions, a full description of the state of an energy-aware system S can be denoted as a three-tuple $s = (n_1, n_2, n_3)$, where n_1 denotes the number of jobs in the system, n_2 denotes the number of servers either *IDLE* or *BUSY* (the number of servers on), and n_3 denotes the number of servers in *SETUP*.

Definition 1. ***Valid State:*** *Given an energy-aware system $S = (N, \lambda, \mu, \gamma)$, a valid state (n_1, n_2, n_3) satisfies $n_1,\ n_2,\ n_3 \geq 0$ and $n_2 + n_3 \leq N$.*

Each of the energy states has a rate of corresponding energy consumption. These rates are denoted $E_{OFF},\ E_{SETUP},\ E_{IDLE}$, and E_{BUSY}. The assumption on the ordering of these rates is as follows: $0 \leq E_{OFF} < E_{IDLE} < E_{SETUP},\ E_{BUSY}$. The non-negativity of E_{OFF} allows for the hibernate/sleep interpretation of the state, but for simplicity E_{OFF} is often assumed to equal 0.

B. Markov Decision Process

A given energy-aware system $S = (N, \lambda, \mu, \gamma)$ can be represented as a Markov decision process (MDP). Without loss of generality we firstly assume S has undergone uniformization. We also use the notation from [21] 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. With uniformization and the above notation in mind, an MDP for an energy-aware system can be defined by (1),

$$\begin{aligned} A_{(n_1,n_2,n_3)} =&\{-n_2, -n_2+1, ..., -1, 0, 1, \\ &..., N-n_2-n_3-1, N-n_2-n_3\}, \end{aligned}$$

and the cost (reward) function $\mathcal{C}((n_1, n_2, n_3), a)$, which for now is left unspecified. The above MDP can also be made interruptible by simply expanding the action space, that is by letting

$$\begin{aligned} A_{(n_1,n_2,n_3)} =&\{-(n_2+n_3), -(n_2+n_3)+1, ..., -1, 0, 1, \\ &..., N-n_2-n_3-1, N-n_2-n_3\}. \end{aligned}$$

III. Main Results and Discussion

As will be seen, the optimal policy for energy-aware systems can in general be quite complex and difficult to determine explicitly. However, there exist several structural properties which the optimal policy satisfies. These properties allow us to make decisions on when to turn servers on and off in a much more intelligent manner. Specifically, some decisions which may appear to be intuitively viable are shown to be suboptimal. This section is primarily reserved for discussion and implications of the results, all proofs are found in Section V.

To understand the more advanced structural properties of the optimal policy, it is first important to understand the rudiments of how the optimal policy behaves. In a Markovian setting like the one we have defined, if an optimal action is performed, then it is immediately following a non-dummy event. This means that it is only optimal to turn off or on a server immediately after a job arrives, a job leaves, or a server finishes its setup. The optimal policy also has two other

$$p((n_1',n_2',n_3')|(n_1,n_2,n_3),a) = \begin{cases} \lambda & \text{if} \quad n_1'+1=n_1,\ n_2'=n_2+\min(0,a), \\ & \qquad n_3'=n_3+\max(0,a) \\ \min(n_1,n_2+\min(0,a))\mu & \text{if} \quad n_1'-1=n_1,\ n_2'=n_2+\min(0,a), \\ & \qquad n_3'=n_3+\max(0,a) \\ (n_3+\max(0,a))\gamma & \text{if} \quad n_1'=n_1,\ n_2'+1=n_2+\min(0,a), \\ & \qquad n_3'-1=n_3+\max(0,a) \\ 1-\lambda & \text{if} \quad n_1'=n_1,\ n_2'=n_2+\min(0,a), \\ -\min(n_1,n_2+\min(0,a))\mu & \qquad n_3'=n_3+\max(0,a) \\ -(n_3+\max(0,a))\gamma & \end{cases} \tag{1}$$

simple structural properties which allow for its analysis to be somewhat simplified before we consider more sophisticated properties.

Theorem 1. *For all energy-aware systems, the optimal policy is always 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 number of non-zero number of servers into setup.*

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

Theorem 2. *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 (n_1,n_2,n_3) it is optimal to begin turning on another i servers, then in state (n_1+1,n_2,n_3) it is optimal to turn on at least i more servers. Likewise when dealing with turning servers off, if it is optimal to turn off i servers in state (n_1,n_2,n_3), then it is also optimal to turn off at least i servers in state (n_1-1,n_2,n_3).

Theorem 2 allows for a convenient description of the optimal policy, by simply providing a set of threshold values. Let $k^+_{i,j}$ and $k^-_{i,j}$ denote the threshold decision variables for an energy-aware system, where $0<i\leq N$ and $0\leq j<i$. The variable $k^+_{i,j}$ is the threshold value to turn on the i^{th} server, while there are j servers turning on. The variable $k^-_{i,j}$ is the threshold value to turn off the i^{th} server, while there are j servers turning on. An initial partial ordering of the threshold values is given from the following definition.

Definition 2. *For all $i,j\geq 0$ such that $i+j<N-1$, $k^+_{i,j}\leq k^+_{i+1,j}$ and $k^+_{i,j}\leq k^+_{i,j+1}$. Furthermore, for all $i>0$ and $j\geq 0$ such that $i+j<N$, $k^-_{i,j}\leq k^-_{i+1,j}$ and $k^-_{i,j}\leq k^-_{i,j+1}$.*

For an energy-aware system, one knows the optimal policy if and only if the values of all of the threshold values are known. Therefore, for an energy-aware system $S=(N,\lambda,\mu,\gamma)$, to find the optimal policy one must determine $N(N+1)$ decision variables. Delving deeper into these threshold values leads to a result which offers a significant reduction to the state space which should be considered.

Theorem 3. *For all $0<i\leq N$ and $0\leq j<i$, $k^-_{i,j}<k^+_{i-1,j}$. Or in other words, if in a valid state (n_1,n_2,n_3-1) it is optimal to begin turning on the $(n_3+n_2)^{th}$ server, then in state (n_1,n_2+1,n_3-1) it is suboptimal to turn off the $(n_2+1)^{th}$ server.*

This theorem is obvious if one were to consider a system with instantaneous setups, i.e. $1/\gamma=0$, since it would make no sense to turn a server on and then immediately turn it off. However, consider a system with long setup times ($1/\gamma$ is large) in some state (n_1,n_2,n_3) where in state (n_1,n_2+1,n_3) it is known that turning a server off is optimal. It may be reasonable to think that although the system were 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 the expected state of the system once the server finishes its setup. However, this is not the case. Furthermore, this gives us a link between the orderings of the off thresholds, and on thresholds. From here a comprehensive partial order for all threshold values can be derived.

Theorem 4. *A partial order can be defined on the threshold values, of which the lattice is shown in Figure 1, where $x\rightarrow y$ denotes $x\leq y$.*

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, then in state $(0,N,0)$ the system would immediately turn off $N-n^*$ servers, where n^* is the number of servers which always remain on. Figure 1 still captures this behaviour, as $k^+_{i,j}\leq 0$ for all $i+j<n^*$, alongside $k^-_{i,j}<0$ for all $i+j\leq n^*$. From Theorem 3 and the fact that there cannot be less than zero jobs in the system, it is known these servers will always remain on.

Theorem 3 offers information on turning servers on with respect to the off thresholds. We would also like to have information about choosing the off threshold directly. 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

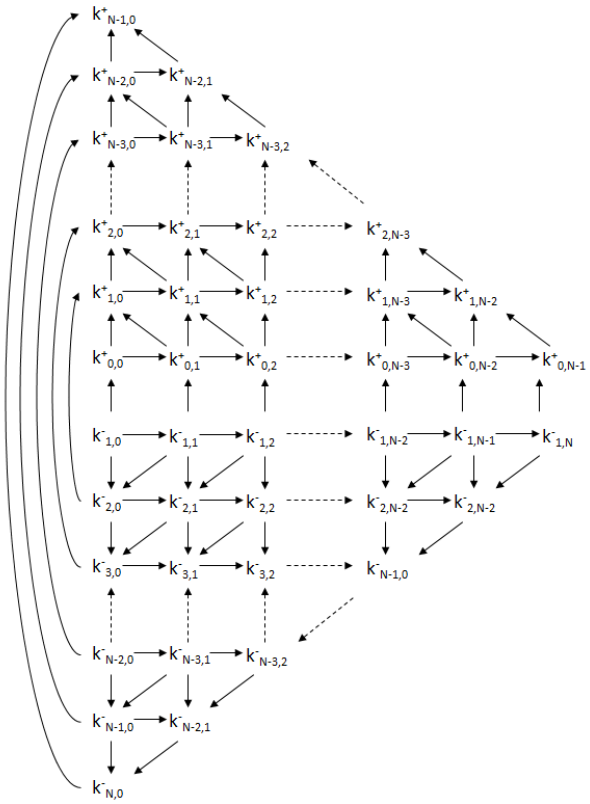

Fig. 1. Threshold lattice: some arrows from Theorem 3 are left off for readability

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 some lower threshold is reached. 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 will cause the system to remove jobs at faster rate. Therefore, other servers could start to turn off sooner. Once those other servers are *OFF*, more jobs could arrive and cause those servers to now start to *SETUP*, incurring a larger energy cost than if the turned off servers just remained on until the new jobs arrived. But if the initial server instead turned off and the system was therefore processing jobs at a slower rate, the other servers could incur a lesser cost by not undergoing additional setup periods. 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, which leads us to our next theorem.

Theorem 5. *Given an energy-aware system $S = (N, \frac{1}{\lambda}, \frac{1}{\mu}, \frac{1}{\gamma})$, where the cost function is only dependent on, increasing and continuous in the expected number of jobs in the system and expected rate of energy consumption, if the number of jobs in the system is greater than or equal to the number of servers currently turned on, it is suboptimal to turn a server off.*

With regards to the faulty intuition presented before Theorem 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 is seen. One of the popular turn off policies used in literature is to turn a server off the moment it idles, such as the "stagger setup" model in [7]. Theorem 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 criteria. 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.

In Section II we described an alteration to our energy-aware system where we allow for the setups to be interruptible. That is, a server is allowed to move directly from *SETUP* to *OFF*. This modification drastically changes the structure of the optimal policy. We define a new class of policies called bulk-setup policies. A bulk-setup policy uses the following setup criteria: for some valid state (n_1, n_2, n_3) if it is better to turn on some number of servers rather than do nothing or turn some number off, then it turns on all available servers, where the number of available servers is given by $N - n_2 - n_3$.

Theorem 6. *For all interruptible energy-aware systems where the cost function is a positive linear combination of the number of jobs in the system and the rate of energy consumed by the system, the optimal policy is a bulk-setup policy.*

While perhaps surprising at first, this type of result arises in other research [3] and follows from the assumption of the exponentially distributed setup times. If you were to turn on k servers, although you would incur the energy costs at k times the rate, due to the nature of the exponential distribution, it would be for an expected amount of time that is reduced by a factor k. 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 N is 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 and off due to potential wear and tear, risk of failure, etc. As such there is a switching cost associated with such an action and some authors incorporate this into their cost function [5], [14], [16]. Clearly, for a 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 functions, 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 [9]. But if one chooses to exclude switching from the cost function and allows their model 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.

IV. Three Server Example

Here we present a toy example with a three server energy-aware system. Define an energy-aware system $S = (3, \lambda, \mu, \gamma)$ where $\lambda < 3\mu$. We do not offer an exact solution to the optimal policy but instead show how theorems from Section III can be applied to constrain the search for the optimal thresholds. Firstly, from Theorems 1 and 2 it is known that there are twelve decision variables. These decision variables are the threshold values $k^+_{i,j}$ where $i, j \geq 0$ and $i + j < 3$, and $k^-_{i,j}$ where $i > 0$, $j \geq 0$ and $i + j \leq 3$.

We begin by examining the off thresholds for the system. In complete ignorance of previously presented results the set of feasible values for these thresholds is $\{-1, 0, 1, 2, ..., K - 1, K\}$ for some constant K. However, applying Theorem 5, arguably the most valuable of our theorems, the feasible state space shrinks significantly. Explicitly the feasible set of values for the off thresholds $k^-_{1,j}$, $k^-_{2,j}$, and $k^-_{3,j}$ 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 a greater confidence in approximations for picking such values, as well as the ability to iterate numerically through a much smaller set of values.

Moving our attention 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 constant, K. While here we have not presented a theorem which can be applied directly without any other knowledge, we can make simplifications assuming we know other threshold values. Specifically, if the off thresholds are known (which as seen earlier can be chosen from a relatively small set of values) then the lower end of the set of feasible values can be truncated by applying Theorem 3. That is for each threshold $k^+_{i,j}$ the feasible set of values now becomes $\{k^-_{i+1,j} + 1, k^-_{i+1,j} + 2, k^-_{i+1,j} + 3, ..., K - 1, K\}$. 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, 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.

V. Proofs of Results

All proofs of the theorems presented in Section III are contained in [15]. However, we consider Theorem 5 to be one of our strongest contributions and therefore offer a proof of it here. Before we proceed with the proof, we must first describe one of our analytical methods.

We wish to reason about finite costs, while still considering the optimal policy under an infinite time horizon. 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 $R_{p,s}$ be a random variable denoting the reward (or cost) earned over a single cycle of state s under policy p. 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 $T_{p,s}$ be a random variable denoting the amount of time it takes to complete a cycle of s under policy p. Given the set of all stable policies $\mathcal{P}$, along with these two random variables, $c(s)$ can be defined as

$$c(s) = \min_{p \in \mathcal{P}} \left\{ \frac{\mathbb{E}[R_{p,s}]}{\mathbb{E}[T_{p,s}]} \right\}.$$

Here $R_{p,s}$ 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 $R_p(t)$ denotes the total reward/cost earned by time t under policy p, then for all states s

$$\frac{R_p(t)}{t} \to \frac{\mathbb{E}[R_{p,s}]}{\mathbb{E}[T_{p,s}]} \text{ as } t \to \infty.$$

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 [11]. Therefore $c(s)$ is independent from s, in other words, $c(s)$ is a constant, which we will denote by c^*. Likewise, the ratio $\mathbb{E}[R_{p,s}]/\mathbb{E}[T_{p,s}]$ is also independent from s and an arbitrary state, say $s = (0, 0, 0)$ can be chosen as the cycle reference. It should be noted that this does not say $\mathbb{E}[R_{p,s}]$ and $\mathbb{E}[T_{p,s}]$ 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 p, $\mathbb{E}[R_{p,s}]/\mathbb{E}[T_{p,s}]$ equals a constant, which we denote by c_p.

To further the usefulness of these cycle costs, an extension $c(s, a)$ is made to $c(s) = c^*$. 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 $\mathcal{P}_{s,a}$,

$$c(s,a) = \min_{p \in \mathcal{P}_{s,a}} \{c_p\}.$$

The values $c(s,a)$ are referred to as the constrained cycle costs. Such a construction is useful for several reasons. Firstly, in any stable energy-aware system for all states s and actions a, unlike the classical MDP optimality equations, $c(s,a)$ is finite. This provides an option that can lead to simpler reasoning when the optimality equations become convoluted. Furthermore, it is also possible (albeit difficult) to arrive at closed form solutions for c^* and $c(s,a)$. Secondly, due to the construction one can say that in state s, a is an optimal action if and only if $c(s,a) = c^*$.

Lemma 1. *For all valid states s and for all valid actions, a, 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 p^*, p^* is not present in the constrained set of policies $\mathcal{P}_{s,a}$. Therefore,

$$c(s,a) = \min_{p \in \mathcal{P}_{s,a}} \{c_p\} > c_{p^*} = c^*$$

which implies $c(s,a) > c^*$ and contradicts the initial assumption. □

A. Proof of Theorem 5

Theorem 5 is one of the more involved proofs offered in this work. The proof uses two lemmas. The first shows that increasing the expected turn on times of an energy-aware system will discourage it from turning a server off, i.e. the optimal threshold to turn a server off is less than or equal to that of a system with shorter setup times. The second determines the optimal policy for an energy-aware system with instant turn on times and shows that the server will never turn off if there is work to be done. The rest of the theorem follows by using these lemmas in conjunction.

Lemma 2. *For all $x > 0$, given two energy-aware systems, $S_1 = (N, \frac{1}{\lambda}, \frac{1}{\mu}, \frac{1}{\gamma})$ and $S_2 = (N, \frac{1}{\lambda}, \frac{1}{\mu}, \frac{1}{\gamma} + x)$, if for S_1, while in state (n_1, n_2, n_3), it is suboptimal to turn a server off, then for S_2 while in state (n_1, n_2, n_3), it is also suboptimal to turn a server off.*

Proof. For the remainder of this proof, the constrained cycle costs for S_1 and S_2 are denoted by $c_1(s,a)$ and $c_2(s,a)$, respectively. To prove the lemma, it is enough to show that given $c_1(s,a_+) < c_1(s,a_-)$, $c_2(s,a_+) < c_2(s,a_-)$, where $a_+ \geq 0$ and $a_- < 0$. An interpretation of this statement is if it is better to either do nothing, or turn a number of servers on rather than turn a number of servers off while in state s for S_1, then this is also true while in state s for S_2.

It is known that for all states and actions, $c_1(s,a) < c_2(s,a)$. This follows from an argument that there exists a policy for the first system which will always incur a lower cost. Consider two systems where all job sizes and job arrival times are equal, they have an equal number of servers, and they share the same reward/cost function. However, the expected setup times in the second system are longer than that of the first. The first system can mimic what the second system does in every way, with the exception of the setup times. If the second system chooses to turn on a server while the corresponding server in the first system is turned off, the server in the first system will also begin to turn on. The associated servers for the first and second system are denoted by v_1 and v_2 respectively. From here, two cases can happen:

- Case 1: v_1 turns on before v_2.
- Case 2: v_2 turns on before v_1.

For Case 1, v_1 can begin processing jobs (or idle if there are no jobs to process) until v_2 turns on. Once v_2 turns on, v_1 will mimic its behaviour while possible, that is v_1 will process jobs while v_2 is processing jobs. If it becomes the case that v_1 has no jobs to process while v_2 does, it will remain idle. Once v_2 is turned off, v_1 is immediately turned off as well and the servers are once again synchronized. A graphical representation of the timeline for Case 1 can be seen in Figure 2, with labels on the times at which key events occur. Using the denotation in Figure 2, one can perform an analysis on the expected holding costs, as well as the expected rate of energy consumption. For ease of argument, if the event at $t_{2,1}$ does not occur, it is assumed that $t_{2,1} = t_{3,1}$. Firstly, it is clear that the first system incurs a lesser holding cost. This is due to v_1 being on while v_2 is still in setup. While the exact decrease in holding cost is unclear, it is known that as long as there are jobs to be processed, the total saved holding costs will increase in time. That is there is some function $h(t)$ which is increasing in t up until $t = t_{2,1} - t_{1,1}$. Therefore, the amount of holding cost saved in the first system is $h(t_{2,1} - t_{1,1}) + x_1$, where x_1 denotes the holding cost of any new job which arrives once v_1 has already idled. Secondly, the rate of energy of consumption is also less in the first system. While v_1 is processing jobs, it is incurring zero energy costs, and when it is idle, it is incurring energy costs at rate less than v_2 by the amount $r_{Setup} - r_{Idle}$. Therefore the total amount of energy saved in the first system is $r_{setup}(t_{2,1} - t_{1,1} + x_2) + (r_{Setup} - r_{Idle})(t_{3,1} - t_{2,1} - x_2)$. Here, x_2 denotes the time it takes to process any other jobs once v_1 becomes idle for the first time, but v_2 has not yet turned on.

For Case 2, v_2 will begin processing jobs (or idling) until v_1 is turned on. Once v_1 does turn on, it will begin processing all available jobs which v_2 had previously processed. Once it completes those jobs, v_1 checks the status of v_2. If v_2 is off, v_1 turns off and the servers are synced. If v_2 is in setup, v_1 will either idle or process jobs if some are available. In the case where v_1 finds v_2 off once the jobs are complete, it will also turn off. A graphical representation for Case 2's timeline can be seen in Figure 3, with labels on the times at which key events occur. Following the upper path of Figure 3 where v_1 inspects v_2 as off once it has completed its work, one can show that the second system has incurred a lesser cost than that

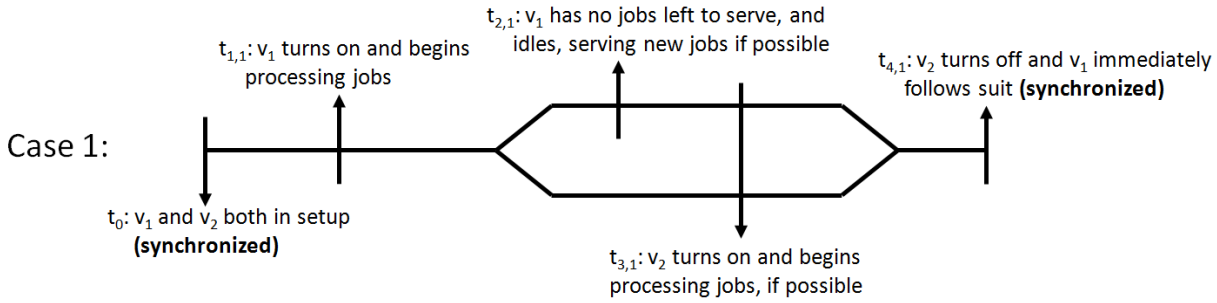

Fig. 2. Case 1 Timeline

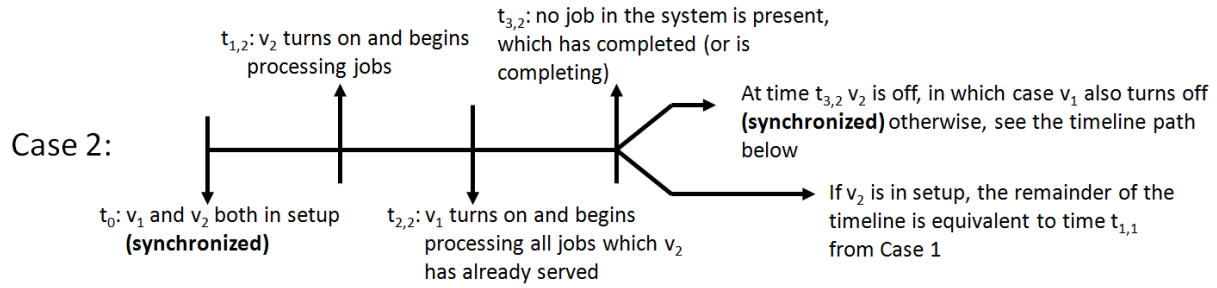

Fig. 3. Case 2 Timeline

of the first. However, it will be shown that the expected cost which the second system saves, is less than the expected cost which the first system saves in Case 1. Furthermore, following the lower path of Figure 3, where v_1 inspects v_2 in setup, it is clear that in this path the second system saves less than it did in the upper path. Therefore if it can be shown that the expected cost saved by the second system in the upper path of Case 2 is less than the cost expected to be saved by the first system in Case 1, weighted by the corresponding probabilities of the cases occurring, then we can conclude that $c_1(s,a) < c_2(s,a)$.

Clearly, the probability of Case 1 occurring once the servers are synchronized is greater than the probability of Case 2 occurring. That is, $\gamma_1/(\gamma_1+\gamma_2) > \gamma_2/(\gamma_1+\gamma_2)$. Furthermore, using the denotation of Figure 3 one can analyse the saved costs in the second system. Following the upper path, the second system will save holding costs. However, the expected saved cost will be less than $\mathbb{E}[h(t_{2,1}-t_{1,1})+x_1]$ (the expected holding cost saved by v_1 in Case 1). This is due to the fact that once on, v_1 and v_2 are homogeneous and process jobs at the same rate. However, the expected amount of time which v_1 would have to process jobs before v_2 turns on, i.e. $\mathbb{E}[t_{1,1} - t_{3,1}] = 1/\gamma_2$, is greater than the amount of time which v_2 would have to process jobs before v_1 turns on, i.e. $\mathbb{E}[t_{1,2} - t_{2,2}] = 1/\gamma_1$. Similarly, it can be shown that the expected energy savings of the second system on this path are also less than the expected energy savings of the first system in Case 1. Because the expected amount of time v_2 has to work before v_1 turns on is less than v_1 has before v_2 turns on and both servers process jobs at the same rate, the expected energy saved in Case 1, must be greater than that saved in Case 2. Since Case 1 has a higher probability of occurring, and in Case 1 the first system has lower cost than the second system in Case 2, it can be concluded that $c_1(s,a) < c_2(s,a)$.

It should be noted that it may seem as though $c(s,a)$ is putting a constraint on the policy which could hinder the above argument, however this is not the case. Limiting the two systems to implement only policies which choose action a in state s, one can instead look at a regeneration process, or complete cycle on state s. Both systems would start in state s and apply action a. Now all the servers in each system are synchronized. From here the previous argument can be applied until the systems return to state s, and the average cost incurred by S_1 will still be less than S_2.

With these notions in mind, consider the following relationship $c_2(s,a) = c_1(s,a) + \mathcal{K}(s,x)$, where $\mathcal{K}(n_1,n_2,n_3,x)$ denotes some extra cost. It is known that $\mathcal{K}(n_1,n_2,n_3,x) > 0$, given the previous observation that $c_1(n_1,n_2,n_3,a) < c_2(n_1,n_2,n_3,a)$. It is also known that the extra incurred cost is solely caused by the decreased setup rate between S_2 and S_1. This is a direct result of the possible extra energy costs, as well as the added holding cost (number in the system, response time, etc.) and/or the increased cycle times that the greater setup times incur. It is important to note that these incurred costs are finite, as the constrained cycle equation must also be finite. Furthermore, is it clear that $\mathcal{K}(n_1,n_2,n_3,x)$ is increasing in x, as longer setup times imply a higher cost under the same policy.

Now consider the constrained cycle costs where the action to turn a number of servers off is forced. Using the given inequality $c_1(s,a_+) < c_1(s,a_-)$, it is known that $c_2(s,a_-) > c_1(s,a_-) \Rightarrow c_2(s,a_-) > c_1(s,a_+)$ Due to the initial assumption, it is known that $c_1(s,a_-) - c_1(s,a_+) = k(s)$, for

some bounded positive value $k(s)$. Since it is also known that $\mathcal{K}(s,x)$ decreases as x decreases, a sufficiently small value for x, say x' can be chosen such that $k(s) \geq \mathcal{K}(s,x')$. Therefore,

$$\begin{aligned} & c_2(s,a_-) > c_1(s,a_-) \\ \Rightarrow \quad & c_2(s,a_-) > c_1(s,a_+) + k(s) \\ \Rightarrow \quad & c_2(s,a_-) > c_1(s,a_+) + \mathcal{K}(s,x') \\ \Rightarrow \quad & c_2(s,a_-) > c_2(s,a_+) \end{aligned}$$

which of course implies that a_- is a suboptimal action, or in other words, it is suboptimal to turn any number of servers off while in state s. This only proves Lemma 2 for $x \leq x'$, while we wish to show it for all $x > 0$. However, we can can repeatedly apply the result. It is important to note that x' is non-zero, as $k(s)$ is bounded positive. Due to the non-zero and positive nature of x', the result can be extended to all positive values of x. □

Lemma 3. *Given an energy-aware system, $S = (N, \frac{1}{\lambda}, \frac{1}{\mu}, 0)$, where the cost function is only dependent on and increasing in the expected number of jobs in the system and expected rate of energy consumption, 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 (n_1, n_2, n_3), $a = min(N - n_3, n_1 - n_3)$.*

Proof. To show this result, it is shown that this policy minimizes both the expected energy used, as well as the expected holding costs. It is known that all jobs must be served eventually otherwise the holding cost of a job can be thought of as infinite, in contrast to the finite energy cost to remove it from the system. Therefore, the lowest energy cost which can be incurred is the energy required to process all the jobs. This is the amount of energy incurred in the policy where $a = min(N - n_3, n_1 - n_3)$, since here a server is never idle, or in setup. Therefore, under the proposed policy, the expected energy costs are minimized.

It is a similar story with the expected holding costs. It is known that the holding costs are minimized when jobs are either served immediately if the number of jobs in the system is less than N, and a job is waiting in the queue if and only if all N servers are busy processing other jobs. This is true in the policy $a = min(N - n_3, n_1 - n_3)$, since due to the instant turn on times, jobs never wait in the queue, unless all servers are currently busy. □

With the above two lemmas proven, we begin the proof of Theorem 5.

Proof. Consider two energy-aware systems $S_1 = (N, \frac{1}{\lambda}, \frac{1}{\mu}, \epsilon)$ and $S_2 = (N, \frac{1}{\lambda}, \frac{1}{\mu}, 0)$. It is assumed that both systems are stable, that is $N\mu > \lambda$. Let p^* be the policy described in Lemma 3. From Lemma 3 it is known that p^* is an optimal policy for S_2. It will first be shown that as $\epsilon \to 0$, p^* is also an optimal policy for S_1. From here, Lemma 2 can be applied to show the desired result.

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 be looking directly at the expected reward/cost and cycle time for the cycle equations of S_1 under p^* and how they compare to the corresponding values in S_2. Since at the moment, individual metrics are what is 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 $c_2^* = \mathbb{E}[E_{2,p^*,s}]/\mathbb{E}[T_{2,p^*,s}]$, where the subscript 2 denotes the values corresponding to S_2, E is a random variable denoting the rate of energy consumption across the cycle of s, $s = (0, n_2^*, 0)$ and n_2^* denotes the number of servers which always remain on. Now if it can be shown that as $\epsilon \to 0$, $\mathbb{E}[E_{1,p^*,s}] \to \mathbb{E}[E_{2,p^*,s}]$ and $\mathbb{E}[T_{1,p^*,s}] \to \mathbb{E}[T_{2,p^*,s}]$ 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.

$$\begin{aligned} \mathbb{E}[E_{2,p^*,s}] \leq \mathbb{E}[E_{1,p^*,s}] \leq \mathbb{E}[E_{2,p^*,s}] + \epsilon\mathbb{E}[N_s] r_{setup} \\ + (1 - \mathbb{E}[P(N_s)])\mathbb{E}[C_E(N_s)|X = 1] \\ + \mathbb{E}[P(N_s)]\mathbb{E}[C_E(N_s)|X = 0], \quad (2) \end{aligned}$$

where N_s is a random variable denoting the number of server setups before completing a cycle of state s in S_2, and $r_{setup} = E_{SETUP}/E_{BUSY}$. 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 $(0, n_2^*, 0)$. These terms represent the cost incurred by 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 at least one 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. Or in other words, S_1 does not complete the cycle where S_2 would have. $C_E(N_s)$ 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, $C_E(N_s)$ must be finite.

There are a few things to note about (2). Firstly, given that at least one extra cycle does not occur, clearly the expected extra energy costs which those cycles add is equal to zero, i.e. $\mathbb{E}[P(N_s)]\mathbb{E}[C_E(N_s)|X = 0] = 0$. Secondly, from inspection, $P(n)$ 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, (2) becomes,

$$\mathbb{E}[E_{2,p^*,s}] \leq \mathbb{E}[E_{1,p^*,s}] \leq \mathbb{E}[E_{2,p^*,s}] + \epsilon\mathbb{E}[N_s]r_{setup} + \left(1 - \frac{1/\epsilon}{1/\epsilon + \lambda}\right)^{\mathbb{E}[N_s]} \mathbb{E}[C_E(N_s)]. \quad (3)$$

From inspection of (3) it can be seen that as $\epsilon \to 0$, $\mathbb{E}[E_{1,p^*,s}] \to \mathbb{E}[E_{2,p^*,s}]$.

To show that the expected rate of energy consumption is continuous in the expected setup times around 0, it remains to show that as $\epsilon \to 0$, $\mathbb{E}[T_{1,p^*,s}] \to \mathbb{E}[T_{2,p^*,s}]$. Furthermore, to show the cost function is continuous in the expected setup times around 0, is must be shown that the expected holding cost of the two systems approach each other as $\epsilon \to 0$. To do this we apply the same methods and arguments as we did to show the continuity of the expected energy consumed over a cycle. A more detailed version of this part of the proof can be found in [15].

Putting this all together it can be concluded that for S_1, as $\epsilon \to 0$, p^* is the optimal policy for any cost function that is increasing and continuous in the expected number of jobs in the system and expected rate of energy consumption. From here it is easy to see that as $\epsilon \to 0$, for all states (n_1, n_2, n_3), where $n_1 \geq n_2$ it is suboptimal to turn a server off. Defining a new energy-aware system $S_3 = (N, \frac{1}{\lambda}, \frac{1}{\mu}, \epsilon + x)$ as $\epsilon \to 0$, one can apply Lemma 2 to say that for all $x \geq 0$, it is suboptimal to turn a server off if there is a job it could be processing. Or in other words, it is suboptimal to turn a server off, if the number of jobs in the system is greater than or equal to the number of servers currently on. □

VI. Conclusion

We presented formalizations for an energy-aware system with N homogeneous servers, where setups may or may not be interruptible. It was shown that these systems can be modelled and analysed as MDPs. Using these MDPs, the optimal policy was shown to be a pure threshold policy with $N(N+1)$ decision variables (threshold values). From here we derived a partial ordering on these threshold values, showed that turning a server off while there is at least one job in the queue is always suboptimal under general cost functions, and showed if setups are interruptible and the cost function is linear in the expected number of jobs in the system as well as the expected energy consumed by the system, then the optimal policy is a bulk setup policy. These structural properties give confidence to previously researched models, give us rules about the optimal policy which help in constructing new models, and provide simplifications to the feasible search space when attempting to arrive at the optimal policy numerically.

Acknowledgement This research was funded by the Natural Sciences and Engineering Research Council of Canada.

References

[1] 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.

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

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

[4] H. Chen, Y. Li, and W. Shi. Fine-grained power management using process-level profiling. *Sustainable Computing: Informatics and Systems*, 2(1):33–42, 2012.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] K. Li. Optimal power allocation among multiple heterogeneous servers in a data center. *Sustainable Computing: Informatics and Systems*, 2(1):13–22, 2012.

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

[15] 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.

[16] 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.

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

[18] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *SIGPLAN Not.*, 44(3):205–216, 2009.

[19] A. Penttinen, E. Hyytia, and S. Aalto. Energy-aware dispatching in parallel queues with on-off energy consumption. In *IEEE International Performance Computing and Communications Conference*, pages 1–8, 2011.

[20] T. Phung-Duc. Exact solutions for M/M/c/setup queues. *arXiv:1406.3084*.

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

[22] J. Slegers, N. Thomas, and I. Mitrani. Dynamic server allocation for power and performance. In *SPEC International Workshop on Performance Evaluation: Metrics, Models and Benchmarks*, pages 247–261, 2008.

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

[24] A. Wierman, L. L. H. Andrew, and M. Lin. *Handbook on Energy-Aware and Green Computing*, chapter Speed Scaling: An Algorithmic Perspective, pages 385–406. CRC Press, 2012.

[25] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM*, 2009.

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