Multi-device Power-saving
An Investigation in Energy Consumption Optimisation

Jean-Marc Andreoli and Guillaume Bouchard
Xerox Research Centre Europe, Grenoble, France

Keywords: Energy Consumption Modelling, Sequential Decision Processes, Exact Multi-dimensional Optimisation.

Abstract: We are interested here in the problem of optimising the energy consumption of a set of service offering devices. Our target is a printing device infrastructure as typically found in a medium or large office, where all the devices are connected to the network, and clients can submit jobs to individual devices through that network. We formulate a cost optimisation problem to find a trade-off between the energy consumption of the infrastructure and the cost of allocating jobs to devices. The latter cost results from the potential gap between the expectation of the clients and the quality of the service delivered by the devices. We present a model of some of the typical constraints occurring in such a system. We then present a method to solve the optimisation problem under these constraints, and conclude the paper with some experimental results.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

1.1 Description of the Problem

We are interested here in the problem of optimising the operation of a set of service offering devices. Our target is a printing device infrastructure as typically found in a medium or large office, where all the devices are connected to the network, and clients can submit jobs to individual devices through that network. In the sequel, we use that target application to illustrate our approach, although it may apply to other contexts as well. Our goal is to design a controller which processes jobs (service requests) and adjusts devices (service providers) in some optimal way which minimises the total cost of operation of the infrastructure. There are two main sources of controllable cost:

- Each device has a specific energy consumption profile which defines several operations modes, each with its own consumption rate and transition cost to other modes. Typically, a device can be either in "working" mode, when it is performing some work for a client (printing), or in one of several "waiting" modes, where it is simply waiting for a new job to arrive. The waiting modes have different energy consumption patterns. We look at the simple case where there are only two waiting modes: `ready` and `sleep`. In `ready` mode, the energy consumption rate is higher than in `sleep` mode, but when a new job is assigned to the device, the wake up energy consumption is lower. To control the energy consumption cost, we assume that the controller has the ability to switch at any time any device in `ready` mode to `sleep` mode.
- Each client is characterised by an assignment utility function which indicates, for each device and job, the adequacy of assigning that job to that device, from the point of view of the client. For example, if the job is in colour and device B is black and white only or does not support the full palette, while device A does, the client estimates the cost incurred by assigning the job to B instead of A. This amounts to quantifying the loss of quality of the printed document, the cost of which depends on the intended use of the document. Similarly, if the job has a high resolution beyond the reach of device B but within that of A (loss of quality, again), or if B is physically much further from the client than A for collection (loss of time). Full elicitation of the utility function of each client is not realistic. Instead, we assume that each job comes together with a cost menu, i.e. a mapping which specifies for each device the client estimate of the cost of assigning that job to that device. To control the assignment cost, we assume that the assignment of each job is performed by the controller when presented the cost menu of the job.

Andreoli J. and Bouchard G..
Multi-device Power-saving - An Investigation in Energy Consumption Optimisation.
DOI: 10.5220/0004042002320237
In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2012), pages 232-237
ISBN: 978-989-8565-21-1
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

- The total cost of operation of the device infrastructure depends on the energy consumption characteristics of the infrastructure on the one hand, on the client assignment utility functions on the other hand, and, of course, on the actual demand, i.e. the sequence of jobs. Demand is intrinsically non deterministic, and may evolve over time, but we assume here that it is governed by some stationary stochastic process which can be learnt from observation over a sufficiently long period of time.

We seek to develop a controller which is in charge of both assigning devices to jobs and dynamically setting timeouts, in such a way as to minimise the total cost of operation, i.e. the sum of the energy consumption cost and the job assignment cost.

1.2 Related Work

Dynamic power management has been the topic of numerous investigations (Benini et al., 2000) and occurs in many contexts: electronic design (Wang et al., 2011), data centre management (Raghavendra et al., 2008; Urgaonkar et al., 2010), wireless sensor networks (Sinha and Chandrakasan, 2001). The problem we tackle in this paper shares many concepts with DPM. As DPM, we seek the optimal control of the energy consumption of a set of hardware devices with controllable energy consumption regimes, under a stochastic flow of service requests, each of which being satisfied by a controllable subset of devices (actually, exactly one device per request in our case). However, unlike many DPM studies, our assumptions derive from a rarely studied, although quite frequent configuration, where requests come together with *soft* allocation constraints in the form of cost menus, and the devices can at any time be dynamically reprogrammed for energy optimisation. Our solution relies on a very generic technique for sequential decision problems, known as Markov Decisions Processes (MDP), widely investigated in the literature (Bertsekas, 2005; Bäuerle and Rieder, 2011), and which is also used by many other DPM solutions. We assume the simplest flavour of MDPs, where the stochastic demand is assumed known in advance, typically learnt from the observation of past sequences. We focus instead on the problem of simultaneously optimising the set of devices at each step of the decision process, allowing the revision of the energy consumption regime of all the devices each time a request is submitted.

2 PROBLEM MODELLING

2.1 Parametrisation

At any time, a device has a *mode* which is either `ready` or `sleep`. Each device k=1:K is characterised by:

- $\overline{a}_k$: cost rate in `ready` mode; $\underline{a}_k$: cost rate in `sleep` mode; we have $\underline{a}_k < \overline{a}_k$ and we let $a_k = \overline{a}_k - \underline{a}_k$;
- $\check{b}_k$: cost of jump from `ready` to `sleep`; $\hat{b}_k$: cost of jump from `sleep` to `ready`;

Let $g_k(x,\tau)$ the overhead cost of maintaining device k in `ready` mode during up to τ time units over a period of x time units. We have:

$$g_k(x,\tau) \quad =_{\text{def}} \quad \mathbb{I}[x<\tau]a_k x + \mathbb{I}[x\geq\tau](a_k\tau + \check{b}_k)$$

The device infrastructure receives job requests in sequence. For the n-th request in the sequence, let $X_n \in \mathbb{R}^+$ be the time elapsed since the previous request; let J_n be the total information available about the associated job (both about its content and its client); and let C_n be the client cost menu which is the K-dimensional vector whose k-th component for k=1:K holds the client estimate of the cost of assigning job J_n to device k.

- We assume that the random variable $O_n = (XJC)_{1:n}$, which captures all that has been observed just after the n-th request, can be deterministically summarised by a single state variable Z_n, called the demand state, which may live in an arbitrary space $\mathcal{Z}$, and which satisfies the following Markov condition

  $$\mathbf{p}(O_n|O_{n-1}) = \mathbf{p}(X_nZ_n|Z_{n-1})\mathbf{p}(J_n|Z_n)\mathbf{p}(C_n|Z_n)$$

  Thus, state Z_n summarises all the information of the past demand which has an impact on the future demand. The choice of a good demand state space $\mathcal{Z}$ and model satisfying the Markov assumptions depends on the characteristics of the actual demand.
- We further assume that the distributions $\mathbf{p}(Z_nX_n|Z_{n-1})$ and $\mathbf{p}(C_n|Z_n)$ are known and stationary (independent of n). We introduce distributions P,Q, which are the drivers of the demand process, as follows[1]:

  $$\mathbf{p}(X_nZ_n|Z_{n-1}) = P(X_nZ_n|Z_{n-1})$$
  $$\mathbf{p}(C_n|Z_n) = Q(C_n|Z_n)$$

[1] P is a joint distribution and we use the same symbol P to denote the marginals and conditionals.

- Finally, we assume that job execution time is negligible. So we do not try to model it, nor the queueing effect which may result from it on devices. This is a natural assumption for the kind of device infrastructures we are targetting, where devices are far from full utilisation and spend most of their time waiting for jobs.

2.2 Formulation as a Sequential Decision Process

The state of the system at any time is given by a pair $\langle \sigma, z \rangle$ where the control state σ is the subset of indices of the devices in `ready` mode and $z \in \mathcal{Z}$ denotes the demand state of the infrastructure. We seek to build a controller which takes as input the stream of job requests, maintains the state of the system and uses it to make decisions at each new job:

- First, the controller must choose the index k of the device to which the job is assigned. We assume here that jobs are immediately assigned upon reception.
- Then, just after the assignment, the controller must choose a family $(\tau_k)_{k\in\sigma}$ of non negative timeouts, where σ is the control state at that time, and each τ_k denotes the sleep schedule for device $k \in \sigma$ i.e. the time after which device k is to be switched to `sleep` mode if no job has been received in between.

Thus, the problem is formulated as the optimisation of a sequential decision process. We consider the optimisation at infinite horizon with discount factor γ. Let $V^!\langle \sigma, z \rangle$ and $V\langle \sigma, z \rangle$ be the optimal cost to go associated with state $\langle \sigma, z \rangle$, respectively before and after an assignment. The optimality equation is given in Figure 1.

- Equation (1) concerns the total cost of assigning, in state $\langle \sigma, z \rangle$, a job with cost menu c to a device k: it consists of the client assignment cost c_k specified in the cost menu, plus, if device k is in `sleep` mode (i.e. $k \notin \sigma$), the wake up cost to lift it to `ready` mode, plus the cost to go after assignment from the new state $\langle \sigma \cup \{k\}, z \rangle$ in which k is now in `ready` mode. The demand state is unchanged because we ignore job execution time, so the job is assumed to be completed immediately after assignment.
- Equation (2) concerns the cost of setting timeouts $(\tau_k)_{k\in\sigma}$ for the devices in `ready` mode in state $\langle \sigma, z \rangle$, when the next job arrives after time x in a demand state z': it consists of the cost $g_k(x, \tau_k)$ of the energy consumption until time x of each device k in `ready` mode with timeout τ_k, plus the discounted cost to go from the new state $\langle \{k \in \sigma | x < \tau_k\}, z' \rangle$ where the control state consists exactly of those devices in σ for which the timeout was not reached at time x (i.e. $\tau_k > x$).

If functions V and $V^!$ satisfy Equations (1) and (2), then the optimal policy for the controller can be formulated as follows:

- When receiving a job with cost menu c in state $\langle \sigma, z \rangle$, assign it to the device k which minimises the minimisation objective of Equation (1).
- Just after assignment in state $\langle \sigma, z \rangle$, set timeouts $(\tau_k)_{k\in\sigma}$ which minimise the minimisation objective of Equation (2).

3 SOLVING FOR OPTIMALITY

We are looking for a solution in $V^!, V$ to the system of Equations (1) and (2). We follow the general procedure of *value iteration*, which alternates updates to $V^!$ from V using Equation (1) and updates to V from $V^!$ using Equation (2). We assume that the demand state space $\mathcal{Z}$ is finite, so the overall state space is finite and both $V^!, V$ can be represented as finite dimension vectors. The update using Equation (1) is quite straightforward: minimisation can be done by enumeration (of the devices), and the integral is turned into a sum, assuming distribution Q is discrete. The update using Equation (2) is more involved, as it requires solving a K-dimensional optimisation. Unfortunately, the optimisation objective has no good properties, such as convexity or smoothness, which would make it amenable to standard optimisation techniques. Furthermore, it is important to reach a global optimum and not just a local one. The rest of this section is devoted to solving the optimisation in Equation (2).

3.1 Transformation of the Objective

Although the optimisation in Equation (2) occurs in the (up to) K-dimensional space of possible timeouts, it can in fact be turned into a sequence of (up to) K uni-dimensional optimisations. To show this, we introduce two side functions $V^\circ\langle t, \sigma, z \rangle$ and $v\langle t, \sigma, z \rangle$ where t is a positive scalar, and σ, z is a state. It can then be shown that the solution in V to Equation (2) can be obtained by solving the system of equations in V°, v, V shown in Figure 2. In that system, all the optimisations in the space of timeouts are captured by a single operator $\downarrow$, defined for any function f on positive scalars by

$$\downarrow f(\tau) = \min_{t \geq \tau} f(t)$$

$$V^!\langle\sigma,z\rangle = \int_c \min_k [c_k + \mathbb{I}[k \notin \sigma]\hat{b}_k + V\langle\sigma\cup\{k\},z\rangle]\mathbf{d}Q(c|z) \quad (1)$$

$$V\langle\sigma,z\rangle = \min_{(\tau_k)_{k\in\sigma}} \int_{xz'} [\sum_{k\in\sigma} g_k(x,\tau_k) + \gamma V^!\langle\{k\in\sigma|x<\tau_k\},z'\rangle]\mathbf{d}P(xz'|z) \quad (2)$$

Figure 1: The optimality equations for the system.

$$V^\circ\langle t,\sigma,z\rangle = \int_{xz'} (-g_\sigma(x,t) + \gamma\mathbb{I}[t\le x]V^!\langle\sigma,z'\rangle)\mathbf{d}P(xz'|z) \quad (3)$$

$$v\langle .,\emptyset,z\rangle = V^\circ\langle .,\emptyset,z\rangle \quad (4)$$

$$v\langle .,\overset{\neq\emptyset}{\sigma},z\rangle = \downarrow((\min_{k\in\sigma} v\langle .,\sigma\setminus\{k\},z\rangle) - V^\circ\langle .,\sigma,z\rangle) + V^\circ\langle .,\sigma,z\rangle \quad (5)$$

$$V\langle\sigma,z\rangle = v\langle 0,\sigma,z\rangle + \int_x g_\sigma(x,0)\mathbf{d}P(x|z) \quad (6)$$

Figure 2: Solving Equation (2) by a sequence of unidimensional optimisations.

In words, this operator returns the greatest monotonic (non decreasing) lower bound of f. It is the only operator in Figure 2 which involves a non discrete optimisation. In more details, we have

- Equation (3) computes a function V° from $V^!$. We use the notation g_σ as a short hand for $\sum_{k\in\sigma} g_k$.
- Then Equations (4) and (5) inductively compute a function v from V°. Eq. (4) yields the values of v at $\sigma = \emptyset$, while Eq. (5) yields the values of v at $\sigma \neq \emptyset$ based on its values at $\sigma\setminus\{k\}$ for each $k\in\sigma$. Hence, function v can be entirely computed in K iterations of updates using initially Eq. (4) and then repeatedly Eq. (5).
- Finally, Equation (6) yields V by taking the values of v at $t = 0$.

The main problem with this transformation is that, while $V, V^!$ can be represented as the finite dimension vectors of their values, since they have a finite domain, that is not the case of V° and v as these functions have an argument which lives in a continuous space. The traditional solution to this problem is to approximate such functions by linear combinations of suitably chosen basis functions. We take a different approach here, which does not rely on approximation.

3.2 Solution Space

Given a demand state z, let $\mathcal{F}^o$ be the *finite* set consisting of the initial functions $V^\circ\langle .,\sigma,z\rangle$ for all possible control state σ. What we are looking for is a function space $\mathcal{F}$ with the following properties:

- *Finite Representability*: functions in $\mathcal{F}$ can be represented by some finite parametric structures;
- *Pointwise Computability*: for any scalar x and function f in $\mathcal{F}$ given by its parametric representation, the scalar $f(x)$ can be computed;
- *Stability* by the operators of Figure 2:

$$(\mathcal{A})\begin{cases} \forall f\in\mathcal{F}^o & f\in\mathcal{F} \\ \forall f,g\in\mathcal{F} & \min(f,g)\in\mathcal{F} \\ \forall f\in\mathcal{F}, g\in\mathcal{F}^o & \downarrow(f-g)+g\in\mathcal{F} \end{cases}$$

A function space satisying all these requirements is called a *solution space*, since it can be used to update exactly V from $V^!$ using the updates given in Figure 2. Now, the main formal result of our study is the following proposition.

Proposition 1. *If $\mathcal{F}^o$ satisfies some minimal conditions, then a solution space can be constructed.*

The conditions on $\mathcal{F}^o$ for Proposition 1 to hold are the following, where $\Delta\mathcal{F}^o$ denotes the set of functions of the form $f-g$ with $f,g\in\mathcal{F}^o$.

- There is a procedure EVAL which, given a scalar t and a function f in $\mathcal{F}^o$, returns the scalar $f(t)$. This includes $t = +\infty$, in which case the returned value is the limit (assumed finite) of $f(t)$ when t tends to infinity. In other words, the initial functions must at least be pointwise computable.
- There is a procedure BEHAVIOUR which, given a function h in $\Delta\mathcal{F}^o$, returns the table of variations of h, i.e. a finite interval partition $(A_i)_{i\in I}$ of $\mathbb{R}^+$, and for each $i\in I$, an indicator $s_i\in\{-1,1\}$ with s_i positive (resp. negative) meaning that h is non decreasing (resp. non increasing) on interval A_i.

When these conditions hold, the solution space promised by Proposition 1 is the function space $\mathcal{F}^*$ defined as follows: $\mathcal{F}^*$ is the space of functions

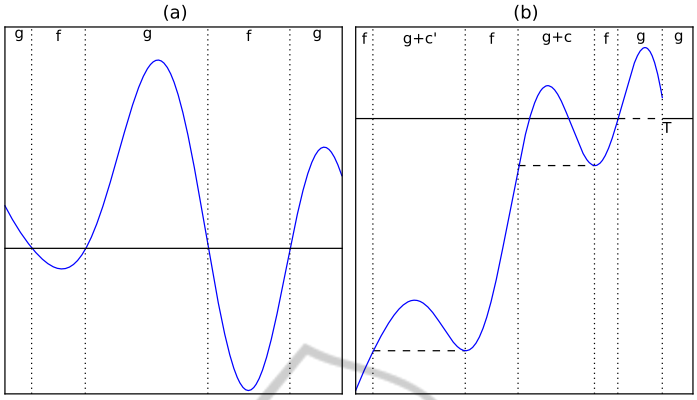

Figure 3: Computing (a): $\min(f,g)$ and (b): $\downarrow(f-g)+g$ given $f-g$ (blue curve); the result (indicated at the top) is piecewise identical to either f or g plus a scalar.

which are piecewise $\mathcal{F}^o$-plus-constant, i.e. of the form $\sum_{i\in I}\mathbf{1}_{A_i}(f_i+r_i\mathbf{1})$ where $(A_i)_{i\in I}$ is a finite interval partition of $\mathbb{R}^+$, $(f_i)_{i\in I}$ is a family of functions in $\mathcal{F}^o$, and $(r_i)_{i\in I}$ is a family of scalars, called the *offsets*. $\mathbf{1}_A$ where A is any interval of $\mathbb{R}^+$ is the characteristic function of A, which returns 1 for the scalars in A and 0 elsewhere. $\mathbf{1}$ is the constant function which always return 1. Let's sketch the proof that $\mathcal{F}^*$ is a solution space:

- $\mathcal{F}^*$ is finitely representable. Indeed, a function in $\mathcal{F}^*$ is entirely described by the bounds of its underlying (finite) interval partition, and for each interval, the corresponding element in $\mathcal{F}^o$ (which is finite) and corresponding offset.
- $\mathcal{F}^*$ is pointwise computable. Given a function f in $\mathcal{F}^*$ and a scalar x, the value $f(x)$ is computed by first determining the interval of the underlying partition of f to which x belongs (this consists in a sequence of comparisons with the bounds of the intervals) and then using procedure EVAL with the function in $\mathcal{F}^o$ corresponding to that interval, and adding its offset.
- Finally $\mathcal{F}^*$ satisfies the stability conditions $(\mathcal{A})$. The first condition is obvious. The intuition for the proof of the other two conditions can be read on Figure 3. Essentially, if f,g are any functions, and $f-g$, represented by the blue curve on the figure, is reasonably well-behaved, then $\min(f,g)$ and $\downarrow(f-g)+g$ are piecewise equal to either f or $g+r\mathbf{1}$ where r is a scalar. If the table of variations of $f-g$ is known, as provided by procedure BEHAVIOUR, then the exact bounds at which $\min(f,g)$ and $\downarrow(f-g)+g$ alternate between the two cases can be computed precisely (as well as the values of r), by simple bi-section[2] on each interval of monotonicity of $f-g$ using procedure EVAL. Technically, what is needed in the proof is the property above for $\downarrow(\mathbf{1}_{[0,T)}(f-g))+g$ for T scalar or infinite, as shown in the figure.

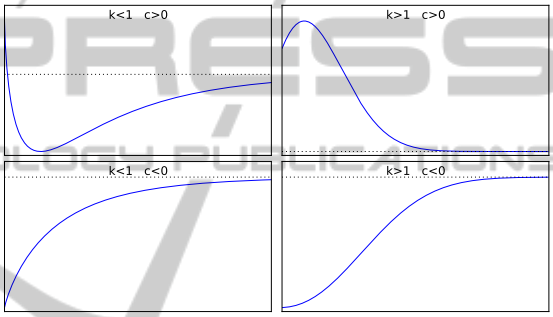

Figure 4: Different shapes of the functions $\psi+c\psi'$ in the case of Weibull demand with shape parameter κ.

3.3 An Example

Consider the case where the demand state space is finite, and, for a given demand state z, distribution P given z is the independent product of a Weibull distribution[3] and a multinomial:

$$\mathbf{d}P(xz'|z)=\pi_{z'}\kappa x^{\kappa-1}\exp(-x^\kappa)\mathbf{d}x$$

where $\sum_{z'}\pi_{z'}=1$. It is easy to show that, up to some scaling factors, the functions in $\mathcal{F}^o$ (and hence those in $\Delta\mathcal{F}^o$) are of the form $\psi(.;z)+c\psi'(.;z)$, where

[2]We mention here bisection, because it is the simplest way to obtain a root of a uni-dimensional function using only a zero-order oracle like procedure EVAL. If higher order oracles are available, then of course more refined methods can be used instead.

[3]Observe that the scale parameter of the Weibull is taken to be 1, thus fixing the time unit.

$$\begin{aligned}\psi(t;z) &= \frac{1}{\kappa}\gamma(\frac{1}{\kappa},t^{\kappa})\\ \psi'(t;z) &= \exp(-t^{\kappa})\end{aligned}$$

For a given c, function $\psi(.;z)+c\psi'(.;z)$ can assume the different shapes, depending on κ and c, shown in Figure 4 and has the following characteristics (when $\kappa \neq 1$):

- It is continuous, has value c at 0, and has a limit at infinity given by $\frac{1}{\kappa}\Gamma(\frac{1}{\kappa})$ where Γ is the complete Gamma function.
- It is monotonically increasing if $c \leq 0$, otherwise it has a single local optimum, at $t^* = (\frac{1}{c\kappa})^{\frac{1}{\kappa-1}}$, which is a maximum if $\kappa > 1$ and a minimum if $\kappa < 1$.
- It is concave if $c \leq 0$ and $\kappa < 1$, otherwise it has a single inflection point solution of the simple polynomial equation $1-\frac{1}{\kappa}+\frac{1}{c\kappa}t-t^{\kappa}=0$.

All this information is sufficient to implement procedures BEHAVIOUR, EVAL, and even higher order oracles than EVAL.

3.4 Experimental Validation

The conditions required for Proposition 1 to hold essentially depend on distribution P in the demand model. They actually hold for a wide range of distributions, and we have derived solutions for various standard distributions (Exponential, Weibull, etc.). We have also conducted experiments both on simulated and real data, which cannot be reported here for lack of space. They are available on demand from the authors. Essentially, in our experiments, we compare the TRANSFER policy obtained by the algorithm proposed here, which minimises the overall cost of the infrastructure, to the SELFISH policy, where each job is assigned the device which minimises only the cost for the client (as given in the cost menu). Unsurprisingly, the TRANSFER policy always performs better, and the gain can be arbitrarily high depending on a characteristic of the demand called the *transfer loss* defined as

$$Q^{+} =_{\text{def}} \min_{z} \int_{c} c^{+} \mathbf{d}Q(c|z)$$

where, for any cost menu c, we let $c^{+} = \min_{k,c_k>0} c_k$ (it is assumed, without loss of generality, that the smallest value in a cost menu is always 0, so c^{+} is the second smallest).

4 CONCLUSIONS

In this paper, we have studied the tradeoff between the client cost of job assignment and the energy consumption of the devices in a framework in which a controller mediates the interaction between client and devices. The kind of system we target is different from the typical jobshop, for which optimisation is a well studied topic. Instead, we target infrastructures in which

- devices spend most of their time waiting for jobs, and the controller can set the energy level at which they do that;
- clients set device assignment constraints, and the controller can override some of them at a price.

The role of the controller is to find a tradeoff between the price of overiding client constraints and the idle energy consumption of the devices. We propose a sequential decision process model of the system as well as a method to achieve the optimal solution.

REFERENCES

Bäuerle, N. and Rieder, U. (2011). *Markov Decision Processes with Applications to Finance*. Springer Verlag.

Benini, L., Bogliolo, A., and De Micheli, G. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on very large scale integration systems*, 8(3):299–316.

Bertsekas, D. (2005). *Dynamic Programming and Optimal Control*. Athena Scientific.

Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., and Zhu, X. (2008). No "power" struggles: Coordinated multi-level power management for the data center. *Operating systems review*, 42(2):48–59.

Sinha, A. and Chandrakasan, A. (2001). Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 18(2):62–74.

Urgaonkar, R., Kozat, U., Igarashi, K., and Neely, M. (2010). Dynamic resource allocation and power management in virtualized data centers. In *Proc. of IEEE/IFIP Network Operations and Management Symposium*, pages 479–486, Osaka, Japan.

Wang, Y., Xie, Q., Ammari, A., and Pedram, M. (2011). Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification. In *Proc. of 48th Design Automation Conference*, pages 41–46, San Diego, CA, U.S.A.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS