Optimal Power-Down Strategies

John Augustine*
jea@ics.uci.edu
School of Information and Computer Science
Univ. of California at Irvine, Irvine, CA 92697

Sandy Irani*
irani@ics.uci.edu
School of Information and Computer Science
Univ. of California at Irvine, Irvine, CA 92697

Chaitanya Swamy†
cswamy@ist.caltech.edu
Center for the Mathematics of Information
Caltech, Pasadena, CA 91125.

Abstract

We consider the problem of selecting threshold times to transition a device to low-power sleep states during an idle period. The two-state case in which there is a single active and a single sleep state is a continuous version of the ski-rental problem. We consider a generalized version in which there is more than one sleep state, each with its own power consumption rate and transition costs. We give an algorithm that, given a system, produces a deterministic strategy whose competitive ratio is arbitrarily close to optimal. We also give an algorithm to produce the optimal online strategy given a system and a probability distribution that generates the length of the idle period. We also give a simple algorithm that achieves a competitive ratio of $3 + 2\sqrt{2} \approx 5.828$ for any system.

1 Introduction

Suppose you are about to go skiing for the first time in your life. Naturally, you ask yourself whether to rent skis or to buy them. Renting skis costs, say, \$30, whereas buying skis costs \$300. If you knew how many times you would go skiing in the future (ignoring complicating factors such as inflation, and changing models of skis), then your choice would be clear. If you knew you would go at least 10 times, you would be financially better off by buying skis right from the beginning, whereas if you knew you would go less than 10 times, you would be better off renting skis every time. Alas, the future is unclear, and you must make a decision nonetheless.

Although the *Ski-Rental* problem is a very simple abstraction, this basic paradigm arises in many applications in computer systems. In these situations, there is a system that can reside in either a low-cost or a high-cost state. Occasionally, it is forced to be in the high-cost state (usually to perform some task). A period between any two such points in time is called an *idle period*.

The system pays a per time unit cost to reside in the high-cost state. Alternatively, it can transition to the low-cost state at a fixed one-time cost. If the idle period is long, it is advantageous to transition to the low cost state immediately; if the idle period is short, it is better to stay in the high-cost state. An online algorithm which does not know the length of the idle period must balance these two possibilities.

This problem has been studied in the context of shared memory multiprocessors in which a thread is waiting for a locked piece of data and must decide whether to spin or block [9, 11]. Researchers investigating

*Research supported partially by NSF grants CCR-0105498 and CCF-0514082 and by ONR Award N00014-00-1-0617.

†Work done while the author was a student at the Department of Computer Science, Cornell University, Ithaca, NY 14853. Research supported partially by NSF grant CCR-9912422.

the interface between IP networks and connection-oriented networks have discovered this same underlying problem in deciding whether to keep a connection open between bursts of packets that must be sent along the connection [12]. Karlin, Kenyon and Randall study the TCP acknowledgment problem and the related Bahncard problem both of which are at heart ski-rental problems [10]. The problem also arises in cache coherency in deciding whether to update or invalidate data that has been changed in a processor's local cache [6, 2].

An important application of the ski-rental problem is in minimizing the power consumed by devices that can transition to a low power *sleep* state when idle. The sleep state consumes less power; however, one incurs a fixed start-up cost in making the transition to the high-power *active* state in order to begin work when a new job arrives. At the architectural level, the technique of eliminating power to a functional component is called clock/power gating. At a higher level, the powered-down component might be a disk drive or even the whole system (e.g., a laptop that hibernates). The embedded systems community has invested a great deal of effort into devising policies governing the selection of power states during idle periods (termed *Dynamic Power Management* in their literature); see, for example, [4] for a survey. These techniques have been critical to maximizing battery use in mobile systems. While power is already a first-class parameter in system design, it will become increasingly important in the future since battery capacities are increasing at a much slower rate than power requirements.

Most of the previous work on this problem has been concerned with two-state systems which have an active state and a *single* sleep state. This paper focuses on finding power-down thresholds for systems that have more than one low-power state. An example of such a system is the Advanced Configuration and Power Interface (ACPI) included in the BIOS on most newer computers, which has five power states, including a hibernation state and three levels of standby [1].

2 Previous work and new results

For the two-state problem, an online algorithm consists of a single threshold T after which time the algorithm will transition from the active to the sleep state. The input to the problem is the length of the idle period and the cost of an algorithm is the total amount of energy it consumes over a single idle period. Typically, an online algorithm is evaluated in terms of its competitive ratio — the ratio of the cost of the online algorithm to the cost of the optimal offline algorithm, maximized over all inputs. When randomized algorithms are considered where the threshold T is chosen at random, we look at the ratio of the expected cost of the online algorithm to the cost of the offline algorithm. Previous work has also addressed the two-state problem when the idle period is generated by a known probability distribution. In this case, the online algorithm will choose a threshold which minimizes its expected cost, where the expectation here is taken over the random choice of the idle period. We call such algorithms *probability-based* algorithms.

The best deterministic online algorithm will stay in the high power state until the total energy spent is equal to the cost to power up from the low power state. It is known that this algorithm achieves the optimal (deterministic) competitive ratio of 2 [9]. When one considers randomized online algorithms, the best competitive ratio achievable improves to $e/(e-1)$ [9]. If the idle period is generated by a known probability distribution, then the algorithm that chooses T so as to minimize the expected cost is always within a factor of $e/(e-1)$ of optimal. Furthermore, this bound is tight since there is a distribution over the idle period lengths which will force any online algorithm to incur an expected cost that is a factor $e/(e-1)$ times larger than that incurred by the optimal offline algorithm [9].

Note that in the context of power-down systems, it may not be the case that the power usage in the sleep state is zero or even that the start-up cost in the active state is zero. In these cases, both the online and the offline algorithm will incur an identical additional cost. Thus, the ratio of the online to the offline cost will decrease and the optimal competitive ratio will be strictly less than two. However, these additional costs do

not change the optimal online or offline strategy in either the deterministic or the probability-based case, and the optimal competitive ratio that can be achieved for such systems can easily be determined as a function of all the parameters of the system.

We denote the problem that involves powering down through k sleep states $PD(k)$. A formal description of the problem is as follows: we are given a sequence of $k+1$ states $S = \langle s_0, ..., s_k \rangle$. There is also a vector of power-consumption rates $\mathcal{K} = \langle \kappa_0, \ldots, \kappa_k \rangle$, where κ_i is the power consumption rate of the system in state s_i. We assume as a convention that the states are ordered so that $\kappa_i > \kappa_j$ for $0 \leq i < j \leq k$. So s_0 is the *active state*, and the system must transition to s_0 (i.e., power up) at the end of the idle period. There is an associated transition cost $d_{i,j}$ to move from state s_i to s_j. A *system* is described by a pair $(\mathcal{K}, d)$. Note that there can be costs to move from high-power states to low-power states and vice versa. However, the only power-up costs that are of interest are the costs to transition from a particular state s_i to the active state s_0 since the only reason to transition to a higher power state is when a new task arrives. A *schedule* or *strategy* $\mathcal{A} = (\mathcal{S}_\mathcal{A}, \mathcal{T}_\mathcal{A})$ consists of a sequence of $n_\mathcal{A} + 1$ states $\mathcal{S}_\mathcal{A}$ that is a subsequence of S, and a sequence of transition times $\mathcal{T}_\mathcal{A}$. Where obvious, we will omit the subscript $\mathcal{A}$. We require that $\mathcal{S}(0) = s_0$ and $T(0) = 0$. We use $\mathcal{A}(t)$ to denote the cost of the schedule produced by strategy $\mathcal{A}$ for an idle period of length t. We also consider a generalization of $PD(k)$ that we call $PD(k, m)$ wherein we require that $n_\mathcal{A} \leq m$, where $0 < m \leq k$ is some limiting integer constant. This generalization would be especially useful for engineers who have a large number of sleep state options available in the design phase, but are required to implement at most a fixed number of states in the product that rolls out into the market.

The only previous work that examines the multiple-state problem $PD(k)$ (from the perspective of worst-case guarantees) is [7] which considers the special case where the cost to power-down is zero and the algorithm only pays to move from low power states to higher power states. Note that this also includes the case where the transition costs are additive ($d_{i,j} + d_{j,k} = d_{i,k}$ for $i < j < k$) since the costs to power down can then be folded into the costs to power up. [7] gives natural generalizations of the algorithms for the two-state case both for the case when the idle period length is unknown and when it is generated by a known probability distribution. It is shown that when the transition costs are additive, the generalized deterministic algorithm is 2-competitive and the probability-based algorithm is $e/(e-1)$-competitive, thus matching the guarantees in the two-state case.

There are two important directions left open by this work. The first is based on the observation that systems, in general, do not have additive transition costs. In many scenarios, additional energy is spent in transitioning to lower power states. Furthermore, there could be overhead in stopping at intermediate states, resulting in non-additive transition costs (see [4] for an example). The second point is that the known upper bounds are typically not optimal *for the system under consideration*. That is, while it is true that there *exist* systems for which the optimal competitive ratio that can be achieved by any deterministic algorithm is 2 (and $e/(e-1)$ by any randomized algorithm), it is possible to achieve a better competitive ratio for many systems. For multi-state systems, the optimal competitive ratio that can be achieved will, in general, be a complicated function of all the parameters of the system (the power consumption rates as well as transition costs). For probability-based algorithms, the optimal competitive ratio will also depend on the probability distribution generating the length of the idle period. While it may not be feasible to express the optimal competitive ratio as a function of all these parameters, a system designer would, in general, like to design a power-down strategy that obtains the best possible competitive ratio given the constraints of his or her particular system.

This paper establishes the following results.

- We give an algorithm that takes as input an instance of $PD(k)$ that is described by $(\mathcal{K}, d)$, and an error parameter ϵ, and produces a power-down strategy $\mathcal{A} = (\mathcal{S}_\mathcal{A}, \mathcal{T}_\mathcal{A})$ whose competitive ratio is within an additive ϵ of the best competitive ratio that can be achieved for that system. The algorithm runs in time $O(k^2(\log k)\log(1/\epsilon))$, where $k+1$ is the number of states in the system, and also outputs the

competitive ratio of $\mathcal{A}$. The algorithm works via a decision procedure which determines for a system and a constant ρ if there is a ρ-competitive strategy for that system. This decision procedure also allows us to obtain lower bounds on the competitive ratio achievable by deterministic algorithms for specific systems, which in turn provides a lower bound on the competitive ratio achievable by deterministic algorithms in general. In particular, we obtain a lower bound of 2.45 on the competitive ratio for deterministic algorithms. This is the first lower bound known that is greater than 2. Independently, Damaschke has given a lower bound of 3.618 [5].

- The above approach can be modified to solve the more general version where a bound of m is specified on the number of states allowed in final strategy. We show how to extend the decision procedure to answer if there is a ρ-competitive strategy for the system that uses at most m power states.

- Experimental results show that there are significant performance gains to be made by estimating the distribution governing the length of an idle period based on recent history and using this estimate to drive a probability-based strategy [8]. We give an algorithm that takes as input a description of a system and a probability distribution generating the idle period length and produces the optimal power-down strategy. Naturally, the running time of the algorithm will depend on the representation of the distribution. In practice, this is most likely to be a histogram. Our algorithm runs in time $O(k^2(\log k + B))$ where B is the number of bins in the histogram and $k+1$ is the number of states. One outcome of the proof is that it also establishes the optimality of the strategy given in [7] for *additive* systems. We then generalize this to find the best online algorithm subject to the restriction that at most m states are used, at the expense of an extra factor of m in the running time.

- We give a simple deterministic strategy that achieves a competitive ratio of $3 + 2\sqrt{2} \approx 5.8284$ for all systems. This result gives a bound on the competitive ratio achieved by the optimal strategies generated by our algorithms. Note that $3 + 2\sqrt{2}$ also serves as a bound on the ratio of the expected costs of the online and offline algorithms when the input is probabilistically generated.

In the remainder of this paper, we use the terms *schedule* or *strategy* interchangeably to refer to the choices of states and threshold times for powering down. The term *algorithm* will refer to a procedure that produces a schedule or strategy based on a particular system.

Azar *et al.* in [3] consider a related problem which they refer to as Capital Investment. This problem is a different generalization of the ski rental problem than the power-down problem considered here. However, a special case of their problem coincides with a special case of our problem. Specifically, they give a $(4 + 2\sqrt{2})$-competitive deterministic algorithm for the special case of the power-down problem in which the cost to transition to each state is the same, regardless of the state from which one is transitioning. Later Damaschke in [5] improves the upper bound on the competitive ratio for this special case (also in the context of Capital Investment) to 4 for deterministic algorithms and 2.88 for ranomized algorithms. In addition, Damaschke gives a 3.618 lower bound for any deterministic algorithm which subsumes the lower bound of 2.45 given here.

3 Preliminaries

First we will establish that we can assume without loss of generality that the power-up transition costs are zero. If this is not the case for some system $(\mathcal{K}, d)$, we can define a new system such that for any $i < j$, the cost to transition from s_i to s_j is $d_{i,j} + d_{j,0} - d_{i,0}$ and the cost to go from s_j to s_i is 0. Since there is never any reason to transition to a higher power state unless the system is transitioning to the active state at the arrival of a new task, any set of actions in the original system will incur the same cost in the new system. Thus, in the sequel we assume that $d_{i,0} = 0$ for all i.

Let $D(i)$ denote $d_{0,i}$. Then $OPT(t) = \min_i(D(i) + \kappa_i t)$. Let $S(t)$ denote the state which attains the minimum — the optimal state. The optimal strategy is to transition to state $S(t)$ at time 0, and stay there through time t. We assume that the optimal strategy will actually "use" every state, i.e., $range(S(t)) = \{s_0, \ldots, s_k\}$. None of the online strategies we present will make use of a state that is never used by the optimal offline strategy for any time t.

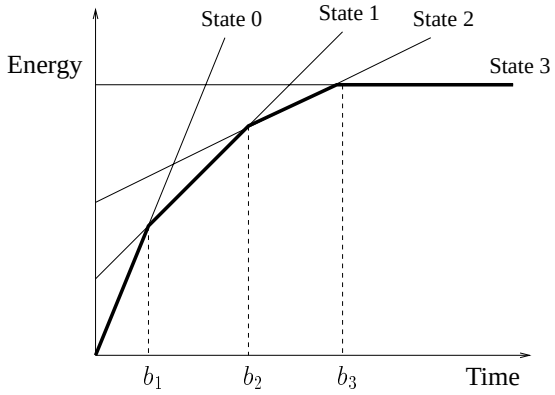

Figure 1: Energy consumed by the optimal strategy as a function of idle period length.

Note that $OPT(t)$ is piecewise linear and $S(t)$ is non-decreasing with t — as the idle period length gets longer, it becomes more worthwhile to pay the extra cost to transition to a lower power state. Let b_i denote the first time instant at which state s_i becomes the optimal state, so $b(0) = 0$ and $D(i-1)+\kappa_{i-1}b_i = D(i)+\kappa_i b_i \Rightarrow b_i = \frac{D(i)-D(i-1)}{\kappa_{i-1}-\kappa_i}$. We have $b(0) < b(1) < \ldots b(k)$. Figure 1 shows the total energy consumed by OPT as a function of the length of the idle period. There is a line for each state. The y-intercept is the transition cost to move to that state from the active state and the slope is the power consumption rate. The energy consumed by the optimal strategy is the lower envelope of these lines since it will pick the single state which minimizes the cost for a given idle period length. Thus for $t \in [b_i, b_{i+1}]$,

$$OPT(t) = D(i) + \kappa_i t = \sum_{j=0}^{i-1} \kappa_j (b_{j+1} - b_j) + \kappa_i (t - b_i) \tag{1}$$

We compare our online strategy with $OPT(t)$ and want to get a strategy $\mathcal{A}$ which minimizes the competitive ratio, $c_{\mathcal{A}} = \sup_t \frac{\mathcal{A}(t)}{OPT(t)}$ where $\mathcal{A}(t)$ denotes the total power consumption of $\mathcal{A}$ by time t.

4 A simple $(3 + 2\sqrt{2})$-competitive strategy

First we establish that we can assume that for all $i < j$, $d_{i,j} < d_{0,j}$. Recall that we are really using $d_{i,j}$ to denote $d_{i,j} + d_{j,0} - d_{i,0}$ and $d_{0,j}$ to denote $d_{0,j} + d_{j,0}$. Thus, the assumption that $d_{i,j} < d_{0,j}$ really amounts to assuming that $d_{i,j} < d_{i,0} + d_{0,j}$. If this were not the case, we could just transition from state s_i to state s_j by first going to s_0 and then down to s_j.

Let us for the moment assume that for some $\gamma > 1$, $D(i) \geq \gamma D(i-1)$ for all $i = 1, \ldots, k$. This is a non-trivial assumption that we will have to handle later. Consider the strategy, $\mathcal{A}$, which always stays in state $S(t)$, the same state as OPT, at every time t. The optimal strategy which knows the length of the idle period in advance will just transition to the optimal state. Strategy $\mathcal{A}$ however must "follow" the optimal strategy, making each transition to a new state as the idle period gets longer. This is the strategy proposed in [7] and shown to be 2-competitive for additive systems. Note that this strategy is the same as the 2-competitive balance strategy for the two-state case.

For $t \in [b_i, b_{i+1}]$ the online cost is, $\mathcal{A}(t) = \sum_{j=0}^{i-1} \big(\kappa_j(b_{j+1} - b_j) + d_{j,j+1}\big) + \kappa_i(t - b_i)$. In comparing this cost to the optimal cost in equation (1), observe that both terms have an additive $\kappa_i(t - b_i)$ which means that the ratio $\frac{\mathcal{A}(t)}{OPT(t)}$ will be maximized at $t = b_i$. To bound the cost of $\mathcal{A}$ in terms of OPT, we use the fact that $OPT(b_i) \leq D(i)$ and $OPT(b_i) = \sum_{j=0}^{i-1} \kappa_j(b_{j+1} - b_j)$ both of which come from equation (1).

$$\begin{aligned}
\mathcal{A}(b_i) &= \sum_{j=0}^{i-1} \Big(\kappa_j(b_{j+1} - b_j) + d_{j,j+1}\Big) \\
&\leq \sum_{j=0}^{i-1} \kappa_j(b_{j+1} - b_j) + \sum_{j=1}^{i} D(j) \\
&\leq OPT(b_i) + D(i) \sum_{j=1}^{i} \gamma^{-(i-j)} \\
&\leq \left(1 + \frac{\gamma}{\gamma - 1}\right) OPT(b_i) \quad = \quad \frac{2\gamma - 1}{\gamma - 1} \cdot OPT(b_i). \qquad (2)
\end{aligned}$$

This holds for any t implying a competitive ratio of $\frac{2\gamma-1}{\gamma-1}$.

Now suppose the assumption $D(i) \geq \gamma D(i-1)$ does not hold. We consider a new *offline* strategy OPT' that only uses a subset of states S' for which the property does hold, and is a γ-approximation of OPT, i.e., $OPT'(t) \leq \gamma \cdot OPT(t)$. We now view our problem as specified by just the states in S', and execute strategy $\mathcal{A}$ as specified above, emulating OPT' instead of OPT. We get that $\mathcal{A}'(t) \leq \frac{2\gamma-1}{\gamma-1} OPT'(t) \leq \frac{\gamma(2\gamma-1)}{\gamma-1} OPT(t)$. Setting $\gamma = 1 + \frac{1}{\sqrt{2}}$, we get a competitive ratio of $3 + 2\sqrt{2} \approx 5.8284$.

We determine OPT' as follows. Let $S' = \{s_k\}$ initially. Consider the states in S in reverse order. Let s_i be the last state added to S'. We find the largest j, $0 \leq j < i$ s.t. $D(j) \leq D(i)/\gamma$. We add s_j to S' and continue until no such j exists. Note that $s_0 \in S'$ since $D(0) = 0$. OPT' will execute the optimal offline strategy assuming that only the states in S' are available. Consider i, j s.t. $s_i, s_j \in S'$ and no s_ℓ is in S' for $i < \ell < j$. We have $OPT'(t) = OPT(t)$ for $t \in [b_i, b_{i+1})$ and $t \in [b_j, b_{j+1})$. For ℓ s.t. $i < \ell < j$ and time $t \in [b_\ell, b_{\ell+1})$. $OPT'(t) = \min(D(i) + \kappa_i t, D(j) + \kappa_j t)$ and $OPT(t) = D(\ell) + \kappa_\ell t$. j was chosen to be the largest value less than i such that $D(j) \leq D(i)/\gamma$ which means that $D(\ell) > D(i)/\gamma$. Furthermore, since $\kappa_i \leq \kappa_\ell$, we have that

$$OPT'(t) \leq D(i) + \kappa_i t \leq \gamma\big(D(\ell) + \kappa_\ell t\big) = \gamma OPT(t),$$

and OPT' is a γ-approximation to OPT.

Theorem 1 *There is a $(3 + 2\sqrt{2})$-competitive strategy for any system.*

5 A near-optimal deterministic algorithm

In this section, we turn our attention to obtaining a near optimal schedule for a particular system. More precisely, given a system $(\mathcal{K}, d)$ with state sequence S for which the optimal online schedule has competitive ratio ρ^*, we give an algorithm that returns a $(\rho^* + \epsilon)$-competitive online schedule in time $O(k^2 \log k \log(1/\epsilon))$. The algorithm is based on a decision procedure which determines whether a ρ-competitive schedule exists for a given value of ρ. Theorem 1 establishes an upper bound of $3 + 2\sqrt{2}$ on the optimal competitive ratio, so we perform a bisection search in the range $[1, 3 + 2\sqrt{2}]$ to find the smallest ρ such that there exists a ρ-competitive schedule. We also output the resulting schedule.

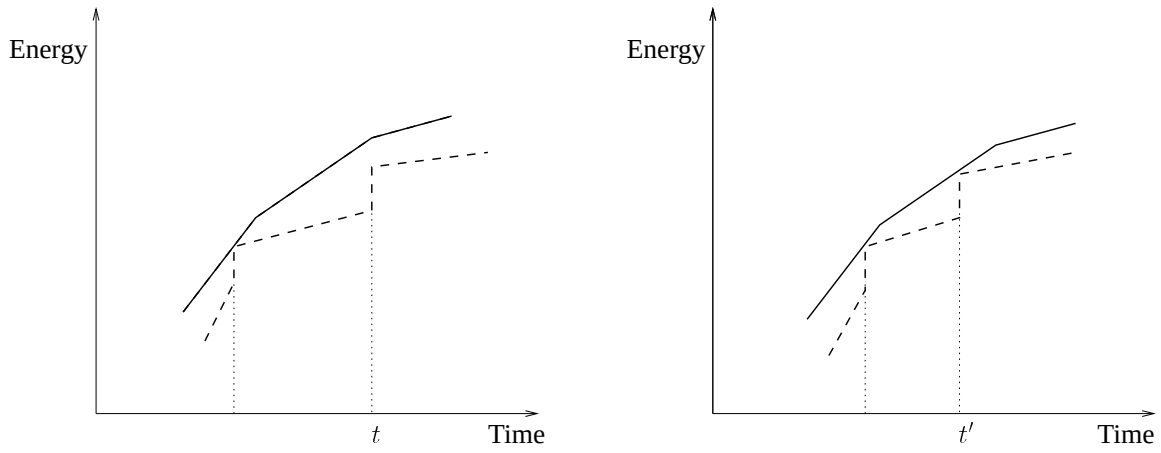

Figure 2: Energy consumed by the online and optimal strategy as a function of idle period length. The solid line is $\rho \cdot OPT(t)$. The dashed line is the online cost. t is the first transition time that is not eager. t' shows the transformed strategy which now has an eager transition.

The following lemma shows that the online strategy must eventually get to a sufficiently low-power state. Lemma 3 allows us to limit our concern to just the transition points in any online schedule.

Lemma 2 *If $\mathcal{A} = (\mathcal{S}, \mathcal{T})$ is a ρ-competitive strategy and s_ℓ is the last state in $\mathcal{S}$, then $\kappa_\ell \leq \rho \cdot \kappa_k$.*

Proof : For the sake of contradiction, assume that $\kappa_\ell > \rho \cdot \kappa_k$. For $\mathcal{A}$ to be ρ-competitive, the function $\mathcal{A}(t)$ must lie entirely below $\rho \cdot OPT(t)$. However the last line of $\rho \cdot OPT(t)$ has slope $\rho \cdot \kappa_k$ and will therefore intersect the last line of $\mathcal{A}(t)$ which has a larger slope κ_ℓ, after which time $\mathcal{A}(t)$ will exceed $\rho OPT(t)$. This is a contradiction. ■

Lemma 3 *If a schedule $\mathcal{A}$ has finite competitive ratio, then the earliest time $\bar{t} > 0$ at which $\frac{\mathcal{A}(t)}{OPT(t)}$ is maximized is a transition point in the strategy $\mathcal{A}$.*

Proof : Let $\rho = \max_{t>0} \frac{\mathcal{A}(t)}{OPT(t)}$. Consider the functions $\mathcal{A}(t)$ and $\rho OPT(t)$. The function $\mathcal{A}(t)$ never exceeds $\rho OPT(t)$, and $\bar{t}$ is the earliest point at which these two functions have the same value, not considering the origin. For the sake of contradiction, assume that $\bar{t}$ is not a transition point in $\mathcal{A}$. So we can find some small $\epsilon > 0$ such that $\mathcal{A}(t)$ is linear in $(\bar{t} - \epsilon, \bar{t} + \epsilon)$. Since $\mathcal{A}(t)$ is strictly less than $\rho OPT(t)$ in the interval $(\bar{t} - \epsilon, \bar{t})$ and $\mathcal{A}(\bar{t}) = \rho OPT(\bar{t})$, it must be the case that the slope of $\mathcal{A}(t)$ is larger than the slope of $\rho OPT(t)$ in this interval. This gives a contradiction, because $\mathcal{A}(t)$ has constant slope over $(\bar{t} - \epsilon, \bar{t} + \epsilon)$, and $\rho OPT(t)$ is a continuous function with decreasing slope, which means that $\mathcal{A}(t) > \rho OPT(t)$ for $t > \bar{t}$. ■

We now explore ways to restrict the space of schedules we need to consider in searching for a ρ-competitive schedule. For a strategy $\mathcal{A} = (\mathcal{S}, \mathcal{T})$, we say that a transition at time $t \in \mathcal{T}$ is *ρ-eager* (or just eager if ρ is clear from the context) if $\mathcal{A}(t) = \rho OPT(t)$. We say that $\mathcal{A}$ is a ρ-eager strategy if $\mathcal{A}(t) = \rho OPT(t)$ for every $t \in \mathcal{T}$. Note that by Lemmas 2 and 3, a ρ-eager strategy that ends at state s such that $\kappa_s \leq \rho \cdot \kappa_k$ is ρ-competitive.

Lemma 4 *If $\mathcal{A} = (\mathcal{S}, \mathcal{T})$ is a ρ-competitive strategy, then there exists an eager strategy $\mathcal{A}' = (\mathcal{S}, \mathcal{T}')$ that is also ρ-competitive.*

Proof : Figure 2 shows a schematic of the proof. The jumps in the online cost (the dashed line) are transition costs. The solid line is $\rho OPT(t)$. The figure shows a transition time t at which the online cost is less than $\rho OPT(t)$. The idea is that we can slide such a transition time earlier until it hits the function $\rho OPT(t)$.

Consider the earliest transition time T which is not eager. Suppose that $\mathcal{A}$ transitions from state s_i to state s_j at time T. Let $T' < T$ be the time of the immediately preceding transition; if there is no such transition time, then set $T' = 0$. The function $\rho OPT(t) - \mathcal{A}(t)$ is continuous in the interval (T', T) since $\mathcal{A}$ does not have any transitions in this open interval, and $\rho OPT(t) - \mathcal{A}(t)$ is 0 at time T' and is strictly greater than $d_{i,j}$ at time $T - \epsilon$ for a small enough ϵ. Let $\overline{T}$ be the earliest time after T' such that $\rho OPT(t) - \mathcal{A}(t) = d_{i,j}$, so $\overline{T} < T$.

Consider the strategy $\mathcal{A}'$ that is identical to $\mathcal{A}$ except that the transition from s_i to s_j is moved earlier from T to $\overline{T}$. we need to argue that $\mathcal{A}'$ is ρ-competitive. Clearly $\mathcal{A}'(t) = A(t)$ for $t \in [T, \overline{T})$ and $\mathcal{A}(\overline{T}) = \rho OPT(\overline{T})$. Also $\mathcal{A}'(T) < \mathcal{A}(T)$ since $\mathcal{A}'$ transitions earlier to the low power state s_j and hence uses less total energy, and since the strategies behave the same after time T, $\mathcal{A}'$ will continue to have a lower cost at all times $t > T$. To see that $\mathcal{A}'(t) \leq \rho OPT(t)$ over the interval $(\overline{T}, T)$, note that $\mathcal{A}'(t)$ is linear over this interval since $\mathcal{A}'$ remains in state s_j. Also $\rho OPT(t)$ is a piecewise-linear concave function since its slope is non-increasing over time. Thus, since the points $(\overline{T}, \mathcal{A}'(\overline{T}))$ and $(T, \mathcal{A}'(T))$ both lie on or below this curve, the straight line connecting them lies under the curve $\rho OPT(t)$.

The procedure above can be repeated until all the transitions are eager. ■

Lemma 5 *Suppose a strategy makes a ρ-eager transition to state s_i at time t_i and next makes a transition to state s_j. Using the function $\rho OPT(t)$, one can compute the earliest ρ-eager transition time $\bar{t}$ to state s_j in time $O(\log k)$.*

Proof : Define the line $l(t) = \kappa_i t + \rho OPT(t_i) - \kappa_i t_i + d_{i,j}$. $\bar{t}$ is the smallest $t > t_i$ such that $\rho OPT(t) = l(t)$. If there is no such t, then a ρ-eager transition from s_i to s_j does not exist. Since $\rho OPT(t)$ is concave we have that if $l(t) < \rho OPT(t)$, or if $l(t) \geq \rho OPT(t)$ and the slope of $\rho OPT(t)$ is less than or equal to κ_i, then $\bar{t} \leq t$; otherwise $\bar{t} \geq t$. These inequalities allow one to do a binary search using the line segments of $\rho OPT(t)$ to determine $\bar{t}$ if it exists. Let s_ℓ be the optimal state (i.e., state of $OPT(t)$) at time t_i. Consider the line segments of $\rho OPT(t)$ corresponding to states s_ℓ and s_k. Recall that b_ℓ and b_k are respectively the left end-points of these segments — these are the first time instants at which s_ℓ and s_k become the optimal states respectively. Using the above inequalities, if we determine that $\bar{t} \geq b_k$, then $\bar{t}$ is simply the point of intersection (if it exists) of $l(t)$ with the segment (of $\rho OPT(t)$) corresponding to s_k. Otherwise we have a "low" segment with end-point b_ℓ, and a "high" segment with end-point b_k. Now we repeatedly consider the left end-point of the segment that is in the middle of the low and high segments, and use the above inequalities to update the low or high segment and the corresponding end-point accordingly, until the end-points of the low and high segments correspond respectively to the left and right end-points of a segment of $\rho OPT(t)$. When this happens we can compute $\bar{t}$ by finding the intersection point (if it exists) of $l(t)$ and this segment. The binary search can be implemented in time $\log k$, where k is the number of segments (i.e., number of states). ■

Lemma 4 immediately gives an algorithm that is exponential in k, the number of states, and determines whether a ρ-competitive strategy exists for the system. This algorithm enumerates all subsequences of states, and determines the ρ-eager strategy for that subsequence by finding the eager transition to each state based on the eager transitions to the previous states, as described in the proof of Lemma 5. A ρ-competitive strategy for the system exists if and only if one of these ρ-eager strategies is ρ-competitive (i.e., ends at a state s with $\kappa_s \leq \rho \cdot \kappa_k$). The remainder of this section presents a way to remove the exponential dependence on k.

Let $S = \langle s_0, s_1, \ldots, s_k \rangle$ be a sequence of states that form a system. Define $S_{s_i \to s_j}$, to be the contiguous subsequence $\langle s_i, \ldots, s_j \rangle$, where s_i and s_j are elements of S such that $i < j$. Let Ψ_s be the set of subsequences of $S_{s_0 \to s}$ that include s_0 and s such that for each $\psi \in \Psi_s$, one can find transition times for the state sequence ψ so that in the resulting schedule, each transition up to and including the transition to state s is a ρ-eager transition. For a state $q \in \psi$, we will use $t_{\psi,q}$ to denote this ρ-eager transition time to q for the sequence ψ. (Note that ψ uniquely determines the transition times $t_{\psi,q}$.)

We define the *earliest* transition time $E(s, \rho)$ of state s for the given system as $E(s, \rho) = \min_{\psi \in \Psi_s} t_{\psi,s}$, that is, $E(s, \rho)$ is the earliest time at which any online strategy can transition to state s while remaining ρ-eager over all its transitions up to (and including) the transition to state s. Observe that if there is ρ-competitive strategy that uses state s, then by Lemma 4, there is such a ρ-eager strategy, so $\Psi_s \neq \phi$ and $E(s, \rho)$ is well defined. We call a transition to state s *ρ-early* (or simply early) if it happens at time $E(s, \rho)$. A strategy that consists entirely of early transitions is called a *ρ-early* strategy.

Lemma 6 *If there is a ρ-competitive strategy $\mathcal{A} = (\mathcal{S}, \mathcal{T})$, then there is an eager and early ρ-competitive strategy.*

Proof : Let s be the last state in $\mathcal{S}$. Consider the sequence $\psi \in \Psi_s$ such that $t_{\psi,s} = E(s, \rho)$ and the strategy π that uses only the states in ψ, transitioning to state $q \in \psi$ at time $t_{\psi,q}$, i.e., $\pi = \big(\psi, \{t_{\psi,q}\}_{q \in \psi}\big)$. Since $\mathcal{A}$ is ρ-competitive, it must be that $\kappa_s \leq \rho \kappa_k$ and since π by definition has all ρ-eager transitions and ends in state s, it is also ρ-competitive. We now argue that π is an early strategy. Note that π was chosen so that the transition to state s is ρ-early. We have to show that the remaining transitions of π are also ρ-early.

Suppose not. Consider the latest transition that is not ρ-early. Suppose this happens for state r ($\neq s$), so $T_1 = t_{\psi,r} > E(r, \rho)$. Let r' be the state just after r in sequence ψ. Let $\psi' \in \Psi_r$ be the sequence for which $t_{\psi',r} = E(r, \rho) = T_2$. T_2 is the earliest time that a ρ-eager schedule can transition to state r and the sequence of states in this schedule is given by ψ'. Consider the hybrid strategy π' that uses the states in ψ' followed by the states in ψ that appear after r, with the transition times being $t_{\psi',q}$ for $q \in \psi'$ and $t_{\psi,q}$ for $q \in \psi_{r' \to s}$. Strategy π transitions to state r at time T_1 and strategy π' transitions to state r at time $T_2 < T_1$. Both of these transitions are eager transitions. Both strategies are in state r at time T_1 and make the same state transitions thereafter. Thus, for any $t \geq T_1$, $\pi(t) - \pi(T_1) = \pi'(t) - \pi'(T_1)$. In particular, both strategies transition to r' (the state after r) at time $t_{\psi,r'} = E(r', \rho) = T'$. Using the equation above we have that $\pi'(T') = \pi(T') - \big(\pi(T_1) - \pi'(T_1)\big)$. We will show that $\pi'(T_1) < \pi(T_1)$ which implies, in particular, that $\pi'(T') < \pi(T')$. So in π' the transition to r' is no longer ρ-eager. Arguing as in Lemma 4 this means that we can shift the transition to r' to get an eager transition at an *earlier* time. But this contradicts the assumption that the transition to state r' at time T' was an early transition.

We now prove that $\pi'(T_1) < \pi(T_1)$. The transitions to state r in schedules π and π' are eager transitions, so both the points $(T_2, \pi'(T_2))$ and $(T_1, \pi(T_1))$ lie on the $\rho OPT(t)$ curve. Since $\pi(t) < \rho OPT(t)$ for all t, the the slope of $\rho OPT(t)$ at time T_1 is at least κ_r, the slope of $\pi(t)$ at time T_1, and strictly greater since the gap between $\rho OPT(t)$ and $\pi(t)$ must accommodate the transition cost from state r to r' at time T'. The concavity of $\rho OPT(t)$ implies that its slope is greater than κ_r over the interval $[T_2, T_1]$. $\rho OPT(t)$. So $\pi(T_1) = \rho OPT(T_1) > \rho OPT(T_2) + \kappa_r(T_1 - T_2) = \pi'(T_1)$ where the last inequality follows since π' stays in state r in the interval $[T_2, T_1]$. ■

From Lemma 6 we can deduce that we only need to consider a specific early and eager schedule, the one that is determined by the $E(., \rho)$ values, to determine if a ρ-competitive strategy exists. We can now define a decision procedure EXISTS that takes a system and a constant ρ and outputs YES if a ρ-competitive strategy exists for the system, and NO otherwise. The procedure can be modified to also output a ρ-competitive strategy (if it exists). We employ a dynamic programming approach to calculate $E(s_i, \rho)$, for $0 < i \leq k$. We always start with the high power state and hence $E(s_0, \rho) = 0$. Suppose we have computed $E(s_j, \rho)$ for all $j = 0, \ldots, i-1$. Let t_j be the earliest time at which the system ρ-eagerly transitions from s_j to s_i given

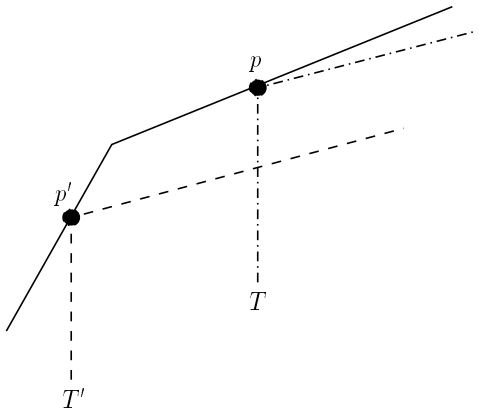

Figure 3: The solid line is $\rho \cdot OPT$. The dashed line is the schedule π' from Lemma 6 and the dashed/dotted line is π. The point labeled p is $(T, \pi(T))$ and p' is $(T', \pi'(T'))$. The idea is to show that at time T, π' has a lower cost than π.

that the transition to s_j is ρ-eager and occurs at time $E(s_j, \rho)$. If such a transition is not possible, then we assign $t_j = \infty$. We can compute t_j in $O(\log k)$ time as described in Lemma 5. Then, $E(s_i, \rho) = \min_{j<i} t_j$. Determining each $E(s_i, \rho)$ requires examining j different possibilities, so finding all the early transition times for all states takes time $O(k^2 \log k)$. By Lemma 2, we know that if $E(s_i, \rho)$ is finite for some state s_i where $\kappa_i \leq \rho \cdot \kappa_k$, we know that a ρ-competitive strategy exists. One can quickly elicit the schedule by starting from state k and retracing the states that minimized the earliest transition time. We use the procedure EXISTS to do a bisection search in the interval $[1, 3 + 2\sqrt{2}]$ and find a ρ-competitive strategy where $\rho \leq \rho^* + \epsilon$. The total time taken is $O(k^2 \log k \log(1/\epsilon))$.

We now turn our attention to adapting this dynamic programming technique to solve $PD(k, m)$ where a bound of m is specified on the number of states that can be used by the online algorithm. We introduce a new parameter b modifying our function to $E(s_i, \rho, b)$, where $0 \leq b \leq \min(i, m)$. The intuition is that function E is now required to return the earliest time when the system can transition to state s_i while staying entirely below $\rho OPT(t)$ and using at most $b + 1$ states from $\langle s_0, \ldots, s_i \rangle$. The base case is $E(s_0, \rho, b) = 0$ for all $b \geq 0$. Intuitively, $E(s_i, \rho, b)$ is determined by the "best" state s_j prior to s_i such that at most $b - 1$ states were used to reach s_j. Notice that for any given state s_i and fixed ρ, $E(s_j, \rho, b)$ is non-increasing as b increases. Therefore, as above we can write $E(s_i, \rho, b) = \min_{j<i} t'_j$, where t'_j is the earliest time when the system ρ-eagerly transitions from s_j to s_i given that the transition to s_j was ρ-eager and occurred at $E(s_j, \rho, b - 1)$. The running time increases by a factor of m now and is $O(k^2 m(\log k) \log(1/\epsilon))$.

6 A probability-based algorithm

Karlin *et al.* study the two-state case when the length of the idle period is generated by a known probability distribution p [9]. (Although they examined the problem in the context of the spin-block problem, their problem is identical to our two-state case.) They observed that the expected cost of the online strategy that makes the transition to the sleep state at time T is

$$\int_0^T p(t)(\kappa_0 t)\,dt + \int_T^\infty p(t)\Big(\kappa_0 T + \kappa_1(t - T) + \beta\Big)\,dt, \tag{3}$$

where κ_0 is the power consumption rate in the active state, κ_1 is the power consumption rate in the sleep state and β is the transition cost between the two states. The online strategy then should select the transition time T that minimizes this cost.

The multi-state case presents two distinct challenges. The first is to determine the optimal sequence of states through which an online strategy should transition throughout the course of the idle period. Then once this sequence has been determined, the optimal transition times need to be determined. Our proof proceeds by establishing that the only transition times that need to be considered are the optimal transition times for two-states systems. Suppose, for example, that we are considering a sequence of state transitions in which state s_i is followed by state s_j. Let $T_{i,j}$ denote the optimal transition time from state s_i to s_j if these were the only two states in the system (that is, if s_i were the active state and s_j were the only sleep state). Note that $T_{i,j}$ can be determined by the expression above. We establish that regardless of the rest of the sequence, the optimal transition point from state s_i to s_j is $T_{i,j}$. We call the $T_{i,j}$'s the *pairwise-optimal* transition times.

Lemmas 7 and 8 establish that the pairwise-optimal transition times happen in the right order. That is for $i < k < j$, $T_{i,k} \leq T_{k,j}$. If this is not the case, then any subsequence that has s_i followed by s_k followed by s_j can not possibly be the best sequence of states. Note that the $T_{i,j}$'s may not necessarily be unique. In general, we will select the earliest transition time that minimizes the cost for the two state system.

Lemma 9 then shows that as long as the pairwise-optimal transition times are in the right order, they give the globally optimal set of transition times for that subsequence. Our algorithm then uses this fact to find the optimal sequence of states by dynamic programming. Note that it is not necessary to exhaustively consider all possible subsequences.

6.1 Optimal transition times

Consider a particular subsequence of $l+1$ states $s_{a_0}, \ldots s_{a_l}$. In order to avoid the double subscripts, throughout this subsection we will rename our subsequence. $q_0, q_1, \ldots, q_l$. Since the strategy must start in state s_0, we can assume that $q_0 = s_0$. For $i < j$, define $\beta_{i,j}$ to be the cost to transition from state q_i to state q_j, that is, $\beta_{i,j} = d_{a_i,a_j}$. Furthermore, we will refer to the power consumption rate of state q_i as α_i, that is, $\alpha_i = \kappa_{a_i}$.

We will consider the strategy that transitions through the states in the subsequence $q_0, q_1, \ldots, q_l$. Suppose that we use transition time T_i to transition from state q_{i-1} to state q_i. It will be convenient for notation to define $T_{l+1} = \infty$ and $T_0 = 0$. The cost of the strategy that uses these transition times is:

$$cost(T_1, \ldots, T_l) \;=\; \sum_{j=1}^{l+1} \int_{T_{j-1}}^{T_j} p(t)\alpha_{j-1}(t - T_{j-1})dt + \sum_{j=1}^{l} \int_{T_j}^{\infty} p(t)\Big(\alpha_{j-1}(T_j - T_{j-1}) + \beta_{j-1,j}\Big)dt. \tag{4}$$

The goal is to pick the $T_1, \ldots, T_l$ so as to minimize the above cost. This is the optimal cost for the subsequence $q_0, \ldots, q_l$.

For each $i \in \{1, \ldots, l\}$, let $\gamma_i = \frac{\alpha_{i-1} - \alpha_i}{\beta_{i-1,i}}$.

Lemma 7 *Suppose that there is an $i < j$ such that $\gamma_i < \gamma_j$, then there is a a strict subsequence of $q_0, \ldots, q_l$ whose optimal cost is no greater than the optimal cost for $q_0, \ldots, q_l$.*

Proof : Consider the first j such that $\gamma_{j-1} < \gamma_j$. Let $(\bar{t}_1, \ldots \bar{t}_{j-1}, \bar{t}_j, \ldots, \bar{t}_l)$ be the sequence of thresholds that minimizes the cost of this sequence of states. Define the following quantities:

$$\begin{aligned} F_{j-1,j} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_{j-1}, \bar{t}_j, \bar{t}_{j+1} \ldots, \bar{t}_l) \\ F_{j-1,j-1} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_{j-1}, \bar{t}_{j-1}, \bar{t}_{j+1} \ldots, \bar{t}_l) \\ F_{j,j} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_j, \bar{t}_j, \bar{t}_{j+1} \ldots, \bar{t}_l) \end{aligned}$$

We will show that $F_{j-1,j}$ is greater than or equal to a weighted average of $F_{j-1,j-1}$ and $F_{j,j}$ which means that it must be greater than or equal to at least one of these values. This means that the strategy that transitions

from state q_{j-2} to state q_{j-1} and then immediately transitions to state q_j at either time $\bar{t}_{j-1}$ or $\bar{t}_j$ is at least as good as the original strategy. Since $\beta_{j-2,j} \leq \beta_{j-2,j-1} + \beta_{j-1,j}$, skipping state $j-1$ altogether can only improve the strategy.

Below we have an expression for $F_{j,j} - F_{j-1,j}$ which can be derived from the definition for the cost in equation (4), Under $F_{j,j}$ the transition from state q_{j-2} to q_{j-1} is moved forward from time $\bar{t}_{j-1}$ to time $\bar{t}_j$. Any time spent in the interval $[\bar{t}_{j-1}, \bar{t}_j]$ happens at the higher power rate of α_{j-2} instead of α_{j-1}. This is accounted for in the first two terms of the sum. However, idle times ending in the interval $[\bar{t}_{j-1}, \bar{t}_j]$ save on the transition cost which is accounted for in the last term below.

$$F_{j,j} - F_{j-1,j} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t - \bar{t}_{j-1})(\alpha_{j-2} - \alpha_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j - \bar{t}_{j-1})(\alpha_{j-2} - \alpha_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \beta_{j-2,j-1} p(t)dt.$$

Dividing by $(\alpha_{j-2} - \alpha_{j-1})$, This becomes

$$\frac{F_{j,j} - F_{j-1,j}}{\alpha_{j-2} - \alpha_{j-1}} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t - \bar{t}_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j - \bar{t}_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \frac{1}{\gamma_{j-1}} p(t)dt. \tag{5}$$

Below, we use the definition of cost in equation (4) to get an expression for $F_{j-1,j} - F_{j-1,j-1}$. Note that in $F_{j-1,j-1}$, the transition from state q_{j-1} to state q is moved back from time $\bar{t}_j$ to time $\bar{t}_{j-1}$. Thus, $F_{j-1,j}$ will spend $\alpha_{j-1} - \alpha_j$ more power than $F_{j-1,j-1}$ for any time spent in the interval $[\bar{t}_{j-1}, \bar{t}_j]$. Furthermore, $F_{j-1,j-1}$ will have an additional transition cost of $\beta_{j-1,j}$ for those intervals that end in the period $[\bar{t}_{j-1}, \bar{t}_j]$.

$$F_{j-1,j} - F_{j-1,j-1} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t - \bar{t}_{j-1})(\alpha_{j-1} - \alpha_j)dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j - \bar{t}_{j-1})(\alpha_{j-1} - \alpha_j)dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \beta_{j-1,j} p(t)dt.$$

Dividing by $(\alpha_{j-1} - \alpha_j)$, This becomes

$$\frac{F_{j-1,j} - F_{j-1,j-1}}{\alpha_{j-1} - \alpha_j} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t - \bar{t}_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j - \bar{t}_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \frac{1}{\gamma_j} p(t)dt. \tag{6}$$

Comparing, equations (5) and (6), the expressions are almost identical except for the γ in the last term. Since $\gamma_{j-1} < \gamma_j$ and $\int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)dt \geq 0$, We have that

$$\frac{F_{j,j} - F_{j-1,j}}{\alpha_{j-2} - \alpha_{j-1}} \leq \frac{F_{j-1,j} - F_{j-1,j-1}}{\alpha_{j-1} - \alpha_j}.$$

Let $\omega_1 = 1/(\alpha_{j-2} - \alpha_{j-1})$ and $\omega_2 = 1/(\alpha_{j-1} - \alpha_j)$. Note that both ω_1 and ω_2 are at least 0. Rearranging, we get that

$$\left(\frac{\omega_1}{\omega_1 + \omega_2}\right) F_{j,j} + \left(\frac{\omega_2}{\omega_1 + \omega_2}\right) F_{j-1,j-1} \leq F_{j-1,j}.$$

■

Now suppose that we consider only the two-state system consisting of state q_{i-1} and state q_i. We will let τ_i denote the optimal threshold time if these are the only two states in the system. We have that τ_i is the time T that minimizes

$$\int_0^T p(t)\alpha_{i-1}t dt + \int_T^{\infty} p(t)\Big(\alpha_{i-1}T + \alpha_i(t-T) + \beta_{i-1,i}\Big)dt.$$

Note that the value of T that results in the minimum above may not be unique. In this case, we take τ to be the smallest value which achieves the minimum. Also note that by subtracting the term $\int_0^\infty p(t)\alpha_i t dt$ (which is independent of T) and dividing by $\beta_{i-1,i}$ in the above definition, it can be seen that $\tau_i = \arg\min_T f(\gamma_i, T)$ where

$$f(\gamma, T) = \int_0^T p(t)\gamma t dt + \int_T^\infty p(t)(\gamma T + 1)dt.$$

Note that for a two-state system whose active state and sleep states has power consumption rates of γ and 0 respectively and whose transition cost is 1, $f(\gamma, T)$ denotes the expected power consumed by an online strategy that transitions to the sleep state at time T. We will show that for a particular subsequence of states, if we minimize the cost over all choices for the thresholds, the resulting thresholds are those obtained by the pair-wise optimization above. First, however, we must establish that the τ_i values have the correct ordering.

Lemma 8 *If $\gamma_i > \gamma_{i+1}$, then $\tau_i \leq \tau_{i+1}$.*

Proof : Intuitively, γ_i is the ratio of the additional power cost of being in state q_i instead of state q_{i-1} over the transition costs between the two states. It stands to reason that the larger this cost, the sooner one would want to transition from state q_{i-1} to state q_i.

We will formalize this argument using a proof by contradiction. Suppose that we have $\tau_i > \tau_{i+1}$ and $\gamma_i > \gamma_{i+1}$. The proof will make use of the definition of $f(\gamma, T)$ given above. τ_i is the smallest value for T which attains the minimum of $f(\gamma_i, T)$. Since $\tau_{i+1} < \tau_i$, we know that $f(\gamma_i, \tau_{i+1}) > f(\gamma_i, \tau_i)$. By the definition of τ_{i+1}, we have that $f(\gamma_{i+1}, \tau_i) \geq f(\gamma_{i+1}, \tau_{i+1})$. Thus, it should be the case that

$$f(\gamma_{i+1}, \tau_i) - f(\gamma_{i+1}, \tau_{i+1}) \geq 0 > f(\gamma_i, \tau_i) - f(\gamma_i, \tau_{i+1}). \tag{7}$$

Using the definition of $f(\gamma, T)$ above, for any $T_1 < T_2$,

$$f(\gamma, T_2) - f(\gamma, T_1) \;=\; \gamma\left[\int_{T_1}^{T_2} p(t)(t - T_1)dt + \int_{T_2}^\infty p(t)(T_2 - T_1)dt\right] - \int_{T_1}^{T_2} p(t)dt.$$

The quantity inside the square braces above is non-negative. This implies that the quantity $f(\gamma, T_2) - f(\gamma, T_1)$ is non-decreasing in γ. This, however, contradicts Inequality 7 and the fact that $\gamma_i > \gamma_{i+1}$. ■

Finally, we prove the main lemma which states that the transition times are simultaneously optimized at the pairwise-optimal transition points.

Lemma 9 *For a given subsequence of states $q_0, \ldots, q_l$, if $\tau_{i-1} < \tau_i$ for all $i \in \{1, \ldots, l\}$, then the minimum total cost is achieved for $cost(\tau_1, \ldots, \tau_l)$.*

Proof : The basic idea is that we can interpret $cost(T_1, \ldots, T_l) - \int_0^\infty p(t)\alpha_l t dt$ as the sum of the power consumed in l two-state systems, where the i^{th} system, (for $i = 1, \ldots, l$), has states whose power consumption rates are $(\alpha_{i-1} - \alpha_i)$ and 0 and the cost to transition between the two is $\beta_{i-1,i}$. Note that $\int_0^\infty p(t)\alpha_l t dt$ is a constant, independent of the choice of T_i's. After rescaling, one can write this expression as a linear combination of the $f(\gamma_i, T_i)$ terms. Since τ_i minimizes $f(\gamma_i, T)$, and the τ_i values have the right ordering, this implies that $cost(T_1, \ldots, T_l)$ is minimized by setting $T_i = \tau_i$ for $i = 1, \ldots, l$.

We will establish below that we can rewrite (4) as follows:

$$cost(T_1, \ldots, T_l) = \int_0^\infty p(t)\alpha_l t dt + \sum_{i=1}^{l}\left[\int_0^{T_i} p(t)(\alpha_{i-1} - \alpha_i)t dt + \int_{T_i}^\infty p(t)\Big((\alpha_{i-1} - \alpha_i)T_i + \beta_{i-1,i}\Big)dt\right]. \tag{8}$$

So by rescaling, we get that

$$cost(T_1, \ldots, T_l) - \int_0^\infty p(t)\alpha_l t dt = \sum_{i=1}^{l} \beta_{i-1,i} f(\gamma_i, T_i).$$

We want to choose $T_1 \leq \cdots \leq T_l$ to minimize this expression. Since $\tau_1 \leq \cdots \leq \tau_l$ and each $\tau_i = \arg\min_T f(\gamma_i, T)$ it follows that the minimum is attained by setting $T_i = \tau_i$ for each i.

To complete the proof we show the equivalence of (4) and (8). It suffices to show that (4) and (8) integrate the same expression over each interval $[T_{i-1}, T_i)$, for $i = 1, \ldots, l+1$. The integrand in (4) over the interval $[T_{i-1}, T_i)$ is

$$p(t)\left[\alpha_{i-1}(t - T_{i-1}) + \sum_{j=1}^{i-1}\Big(\alpha_{j-1}(T_j - T_{j-1}) + \beta_{j-1,j}\Big)\right],$$

and the integrand in (8) is

$$p(t)\left[\sum_{j=1}^{i-1}\Big((\alpha_{j-1} - \alpha_j)T_j + \beta_{j-1,j}\Big) + \Big(\sum_{j=i}^{l}(\alpha_{j-1} - \alpha_j) + \alpha_l\Big)t\right]. \tag{9}$$

The summations over the j indices in (9) telescope to show that the two expressions are identical. ■

6.2 The optimal state sequence

We now present a simple polynomial time algorithm to obtain the optimal state sequence for a given system. First, for each pair $(i, j), 0 \leq i < j \leq k$, let $T_{i,j}$ denote the optimal transition point if s_i and s_j were the only two states in the system. The time complexity of determining a single $T_{i,j}$ depends on the representation of the probability distribution. In practice, this is most likely to be estimated by a finite histogram with B bins starting at time 0 and sampled at a uniform discrete interval of δ. It follows that bin i corresponds to time δi. It is not difficult to generalize this for variable sized bins. We will also assume that all transition times occur at some δi. The height of bin i is $H(i)$ and this implies that the probability that the idle time t equals δi is given by $\frac{H(i)}{\sum_i H(i)}$. In Algorithm 1, we calculate $ACC[i]$ and $ACCT[i]$ values, which are $\int_0^{i\delta} p(t)dt$ and $\int_0^{i\delta} tp(t)dt$ and we then use them to evaluate $T_{i,j}$ values. We can re-write the expression for the cost of a two state system in equation (3) as

$$\kappa_i \int_0^T p(t)tdt + \kappa_j \int_T^\infty p(t)tdt + \Big((\kappa_i - \kappa_j)T + \beta_{i,j}\Big)\int_T^\infty p(t)dt.$$

We also denote $\int_0^{B\delta} p(t)dt$ and $\int_0^{B\delta} tp(t)dt$ as $TOTAL$ and $TOTALT$ respectively. Using the pre-calculated values above, the cost of transitioning from state s_i to state s_j at time δl is

$$\kappa_i \cdot ACCT[l] + (\kappa_i l\delta - \kappa_j l\delta + \beta_{i,j})(TOTAL - ACC[l]) + \kappa_j(TOTALT - ACCT[l]).$$

Once the $T_{i,j}$'s are found, we sweep through them in non-decreasing order, keeping a running tab of the best sub-schedules that we can achieve ending in each state s_i at each point in time. When we encounter a $T_{i,j}$, we check to see if transitioning from s_i to s_j can improve the current best sub-schedule ending in s_j, and if it does, update our data structure to reflect it.

A given strategy divides time into intervals where each interval is the period of time spent in a particular state. The expected cost for a strategy given in equation (4) is obtained by summing over the expected cost

Algorithm 1 Evaluating $T_{i,j}$ values

$ACC[0] \leftarrow H[0]$
$ACCT[0] \leftarrow 0$
for $k = 1$ to B **do**
  $ACC[k] \leftarrow ACC[k-1] + H[k]$
  $ACCT[k] \leftarrow ACC[k-1] + H[k] \times k \cdot \delta$
end for
$TOTAL \leftarrow ACC[B]$
$TOTALT \leftarrow ACCT[B]$
for all (i, j) pairs such that $0 \leq i < j \leq k$ **do**
  $min \leftarrow \infty$, $argmin \leftarrow -1$
  for $l = 0$ to $B - 1$ **do**
    $val = \kappa_i \cdot ACCT[l] + (\kappa_i l\delta - \kappa_j l\delta + \beta_{i,j})(TOTAL - ACC[l])$
        $+\kappa_j(TOTALT - ACCT[l])$
    if $val < min$ **then**
      $min \leftarrow val$
      $argmin \leftarrow l$
    end if
  end for
  $T_{i,j} \leftarrow argmin \cdot \delta$
end for

incurred in each interval. The cost for each interval is divided into two parts which results in two separate summations in equation (4). We define the function Q for the first term which is

$$Q(t_s, j, t_f) = \int_{t_s}^{t_f} p(t)\kappa_i(t - t_s)dt.$$

This is the expected cost of staying in state s_i in the interval $[t_s, t_f)$ for those idle periods whose length is also in the interval $[t_s, t_f)$. Define

$$R(i, t_s, j, t_f) = \int_{t_f}^{\infty} p(t)\Big(\alpha_i(t_f - t_s) + \beta_{i,j}\Big)dt.$$

This is the expected cost for those intervals longer than t_f of staying in state s_i over the time period $[t_s, t_f)$ and then transitioning to state s_j. Note that $Q(\delta l_i, j, \delta l_j)$ and $R(i, \delta l_i, j, \delta l_j)$ can both be evaluated in constant time given $ACC[l_i]$, $ACC[l_j]$, $ACCT[l_i]$ and $ACCT[l_j]$ defined above.

At each transition $T_{i,j}$, we check to see if the current best schedule that ends in state s_j can be improved by transitioning to j from the current best schedule that ends in state s_i. For this purpose, we maintain two arrays of size $k + 1$: $t[i]$ is the time at which the current best schedule that ends at state s_i transitions to s_i and $h[i]$ is the cost at $t[i]$ of that schedule. Initially, $h[0] \leftarrow 0$ and all other $h[i] \leftarrow \infty$. $t[i]$, for all i can be initialized to 0. In Procedure 2, we provide the pseudocode for processing at each transition point $T_{i,j}$.

It is easy to see that each transition point takes a constant amount of processing. The sorting takes an overhead of $O(k^2 \log k)$. The initial preprocessing to calculate the transition points takes $O(k^2 B)$. Hence, the total running time is $O(k^2(\log k + B))$.

The algorithm can be easily extended to find the algorithm that minimizes the expected cost subject to the constraint that only m states are ever reached. We maintain $t[i, b]$ and $h[i, b]$ for all states s_i and $b < \min\{m, i\}$. These are the best time and energy required to reach state i subject to at most b states being reached. The algorithm is given below in Procedure 3.

Procedure 2 Processing $T_{i,j}$ in the line sweep algorithm

Current Status: $T_{i,j}$ is the transition point that is being processed
{The cost up to time $T_{i,j}$ if transitioning from i to j at $T_{i,j}$}
$h1 \leftarrow h[i] + Q(t[i], j, T_{i,j}) + R(i, t[i], j, T_{i,j})$
{The cost up to time $T_{i,j}$ if transitioning to j at the current best time of $t[j]$}
$h2 \leftarrow h[j] + Q(t[j], j, T_{i,j}) + R(j, t[j], j, T_{i,j})$
if $h1 < h2$ **then**
  $h[j] \leftarrow h1$
  $t[j] \leftarrow T_{i,j}$
end if

Procedure 3 Processing $T_{i,j}$ in the line sweep algorithm with the number of states constrained

Current Status: $T_{i,j}$ is the transition point that is being processed
for $b = 1 \ldots j-1$ **do**
  $h1 \leftarrow h[i, b-1] + Q(t[i, b-1], j, T_{i,j}) + R(i, t[i, b-1], j, T_{i,j})$
  $h2 \leftarrow h[j, b] + Q(t[j, b], j, T_{i,j}) + R(j, t[j, b], j, T_{i,j})$
  if $h1 < h2$ **then**
    $h[j, b] \leftarrow h1$
    $t[j, b] \leftarrow T_{i,j}$
  end if
end for

References

[1] `http://www.microsoft.com/windows2000/techenthusiast/features/standby1127%.asp`.

[2] C Anderson and A Karlin. Two adaptive hybrid cache coherency protocols. In *Proceedings of the Second International Symposium on High-Performance Computer Architecture*, pages 303–313, 1996.

[3] Y. Azar, Y. Bartal, E. Feuerstein, A. Fiat, S. Leonardi, and A. Rosen. On capital investment. *Algorithmica*, 25:22–36, 1999.

[4] L. Benini, A Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, 8(3):299–316, 2000.

[5] Peter Damaschke. Nearly optimal strategies for special cases of on-line capital investment. *Theoretical Computer Science*, 302:35–44, 2003.

[6] S. J. Eggers and R. H. Katz. Evaluating the performance of four snooping cache coherency protocols. In *Proceedings of the 16th annual international symposium on Computer architecture*, pages 2–15. ACM Press, 1989.

[7] S. Irani, R. Gupta, and S. Shukla. Competitive analysis of dynamic power management strategies for systems with multiple power savings states. In *IEEE Conference on Design, Automation and Test in Europe*, 2002.

[8] Sandy Irani, Sandeep Shukla, and Rajesh Gupta. Online strategies for dynamic power management in systems with multiple power saving states. *Trans. on Embedded Computing Sys.*, 2003. Special Issue on Power Aware Embedded Computing.

[9] A Karlin, M. Manasse, L. McGeoch, and S. Owicki. Randomized competitive algorithms for non-uniform problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 301–309, 1990.

[10] Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic tcp acknowledgement and other stories about e/(e-1). In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 502–509, 2001.

[11] Anna R. Karlin, Kai Li, Mark S. Manasse, and Susan Owicki. Empirical studies of competitve spinning for a shared-memory multiprocessor. In *Proceedings of the thirteenth ACM symposium on Operating systems principles*, pages 41–55. ACM Press, 1991.

[12] S Keshav, C Lund, S Phillips, N Reingold, and H Saran. An empirical evaluation of virtual circuit holding time policies in ip-over-atm networks. *IEEE Journal on Selected Areas in Communications*, 13(8):1371–1382, 1995.