Online Learning Algorithms for Dynamic Power Management

Zhen Ma Rajesh Gupta
Department of Computer Science & Engineering
University of California at San Diego, La Jolla, CA 92093
{zhma,gupta}@cs.ucsd.edu

Abstract

Dynamic Power Management (DPM) is a major technique to reduce energy consumption for battery-operated embedded systems. Online DPM algorithms refer to strategies that switch the system to optimal *power state according to the idle period lengths at runtime. In this paper, we propose an online learning based power management technique, which combines a low-overhead stochastic learning automaton and threshold based DPM algorithms. The simulation results show that the hybrid algorithm can achieve on an average of 5% and up to 20% energy savings than the online probability-based approach while introduces on an average of 14% lower average latency than any other algorithm that has similar energy savings.*

1 Introduction

Energy efficiency is a major concern in battery-operated real-time embedded systems. Power management can be achieved by switching the system components (such as hard disks, wireless interface, embedded processors, etc.) into power states that are sufficient to meet their performance requirements. When the system is idle, it can be put into a low power state. However, a wakeup energy and latency are required to transition the system back to the active power state. Based on the characteristics (power dissipation, wakeup energy/latency overhead) of each power state and the observation of the idle period lengths, DPM algorithm should make optimal power state transition decisions to minimize the total energy consumption under performance constraints.

An offline DPM algorithm which knows the idle period length in advance can determine whether the length of idle period is long enough such that the power savings can outweigh the wakeup energy overhead and thus choose an optimal state for the idle period. On the contrary, online dynamic power management should make critical decisions without knowing the lengths of upcoming idle periods. In [5], Hwang *et. al.* use the weighted exponential average of the previous idle period lengths to predict the upcoming idle period length. Chung *et. al.* [1] propose a prediction scheme for multiple power state systems, which applies an adaptive learning tree to guide power state transition decisions. These prediction algorithms were categorized as single-value prediction algorithms in [4, 6], for they predict the next idle length as a single value to guide the power management. However, such algorithms cannot capture the nondeterministic feature of the upcoming idle periods. The corresponding state decided by the predicted single value may have a high penalty probability. Markov decision process (MDP) based power management were proposed in [9, 12, 11] to derive optimal power management policy, which is more applicable for offline policy generation due to its high optimization overhead and its limitation in that it assumes the workloads follow predefined distributions. Chung *et. al.* [2] propose an online version of this approach which switches between the interpolations of precomputed offline policies, but this assumes the interpolation can well approximate the optimal DPM policy.

In order to find a better DPM policy that utilizes the statistical information from previous workload, Irani *et. al.* [6] propose an Online Probability-Based Algorithm (OPBA) which learns the idle length distribution and periodically updates the thresholds for power state transitions to minimize the expected energy consumption. Kachroo *et. al.* [7] introduce Stochastic Learning Automaton (SLA) for DPM. An SLA uses the current probabilities for each power state to make state transition decision, evaluates the decision at a later time and update the probabilities by reinforcement learning schemes to learn from its previous mistakes.

Most of the online DPM algorithms are based on the fact that the upcoming idle period length or the optimal decision for the upcoming idle period is most relevant to the recent history. The efficiency of the online DPM algorithms depends on how well it can utilize the idle period information in the histogram. In this paper, we apply various reinforcement learning algorithms to improve online decision making for power state transitions with low computa-

tional overhead. We also propose a hybrid online learning algorithm which combines a stochastic learning automaton with dynamic thresholds computed by OPBA. To understand and extensively explore the algorithmic aspects, we use real hard disk traces to compare our proposed approach with existing DPM algorithms. We then show how the proposed approach is robust across variations in the workloads and can be effective in practice. With this new approach, we achieve on an average of 5.4% and up to 20% energy savings than OPBA. In addition, the hybrid algorithm introduces on the average of 14% lower average latency than the other algorithms that has similar energy savings.

The rest of the paper is organized as follows: Section 2 summarizes the Online Probability-based Algorithm and the Stochastic Learning Automata approach and points out their limitations respectively. Section 3 proposes the hybrid learning approach and the associated SLA design. The experimental results are presented in Section 4, followed by a brief conclusion in Section 5.

2 Dynamic Power Management by Online Learning

In this section, we first present the system model for DPM. We then summarize two best existing DPM algorithms in the literature by online learning and point out their limitations, which motivate our proposed algorithm.

2.1 System Model

Consider a system that has $k+1$ power states denoted by $S_0, \cdots, S_k$. The power consumption for state i is denoted by α_i, where $\alpha_i > \alpha_j$ given $i < j$. For simplicity, we assume power-up is only allowed to the active power state (S_0) while it is allowed to transition to any lower power state from any power state. The total energy consumed in transitioning from S_i to S_0 is denoted by β_i, where $\beta_i < \beta_j$ given $i < j$. In case the time and energy overhead incurred in transitioning to lower power states is non-negligible, they can be incorporated by folding them into the corresponding power-up parameters [6]. A power manager is a controller which observes the idle period lengths and runs the DPM algorithm to make power state transition decisions. For each DPM algorithm, we should consider three factors: energy consumption, average latency and implementation overhead. The energy consumption is our objective metric for energy minimization, while the average latency is the latency introduced by DPM. A more aggressive DPM algorithm usually introduces more latency to the system due to its aggressiveness to transition the system to low power states. Thefore, DPM is a tradeoff between energy and latency. In addition, the DPM algorithm must be implemented efficiently. A complicated DPM algorithm may make more accurate predictions and thus make better tradeoff for energy and latency, but it usually has higher computational and space complexity than a simple DPM algorithm and thus introduces overhead on the power manager itself.

2.2 Online Probability-based Algorithm

Irani *et. al.* [6] introduce an probabilistic analysis approach that models the upcoming input sequence by a probability distribution that is learnt based on historical data. Assume a system has $k+1$ power states denoted by $S_0, \cdots, S_k$. The power consumption for state i is denoted by α_i. Assume the low power states only transition to the active power state (S_0) when it receives a new service request. The total energy consumed in transitioning from S_i to S_0 is denoted by β_i. Assume the idle interval is generated by a fixed, know distribution whose density function is π. τ_i ($i \in \{1, 2, \cdots, k\}$) is the threshold at which time the system will transition from S_{i-1} to S_i. The expected energy cost for the algorithm is given as:

$$\sum_{i=0}^{k} \int_{\tau_i}^{\tau_{i+1}} \pi(t)[\alpha_i t + \beta_i]dt \tag{1}$$

This approach is called Probabilistic Lower Envelope Algorithm (PLEA), which assumes the workload is known a priori. PLEA makes no assumption about the form of the learnt workload distribution and is proved that for any distribution, the expected cost of PLEA is within a factor of $\frac{e}{e-1}$ of the expected cost for the optimal offline algorithm.

The online version of this approach is called Online Probability-Based Algorithm (OPBA). OPBA defines a learning window which is used to summarize the workload history to a histogram. The histogram is then used to generate a new DPM policy periodically in a certain frequency. Instead of using a continuous probability distribution π in PLEA, OPBA uses a discrete probability distribution as the follows: (1) Divide the idle length range $[0, \infty)$ to n intervals. Let r_i be the left endpoint of the i^{th} interval. (2) Associate each interval with a counter c_i which records the number of idle periods in the history window (with a size of w) whose length fall into interval $[r_i, r_{i+1})$. (3) The probability the idle period has length r_i is c_i/w. The threshold τ_i is taken to be

$$argmin_{r_t}\{\sum_{j=1}^{t-1}(\frac{c_j}{w})r_j(\alpha_i-\alpha_{i-1})+\sum_{j=t}^{n}(\frac{c_j}{w})[r_t(\alpha_i-\alpha_{i-1})+(\beta_{i-1}-\beta_i)]\} \tag{2}$$

OPBA provides an efficient way to solve the online DPM problem in that it dynamically updates the transition thresholds based on the idle length distribution. eHowever,

(1) The optimal window size and the threshold update frequency are time-dependent on the actual non-stationary

workload. With a fixed window size, the history information has same weight on each entry in the histogram. The algorithm may not adapt to bursty behaviors in service requests distribution, where old history should decay quickly enough to calculate optimal thresholds.

(2) Although OPBA may converge to a low threshold value when the idle periods in the history are longer than the break-even times, similar to other threshold based techniques, OPBA wastes the energy before reaching the thresholds. It is proved in [3] that it is always more costly to transition through a sequence of power states than to immediately switch to the lowest power state in the sequence for any idle period length. Therefore it may be more effective to shutdown the system or transition to a deep low power state immediately when the system becomes idle.

(3) Threshold candidates are fixed. The granularity of threshold optimization should tradeoff with computation overhead of the algorithm.

2.3 Stochastic Learning Automata

A stochastic learning automaton is a finite state machine that interacts with a stochastic environment and tries to learn the optimal action offered by the environment via a learning process. An SLA can be used as a decision tool: the automaton selects one of the possible states according to a state probability matrix; the chosen action will trigger the environment to response with a feedback. Based on the feedback, the automaton will update the state probability matrix by means of a reinforcement learning scheme.

An SLA can be described precisely in terms of following entities [8]:

- The state of the automaton at any instant t, denoted by $\phi(t)$, is an element of the finite set

$$\Phi = \{\phi_1, \phi_2, ..., \phi_s\} \tag{3}$$

- The output or action of an automaton at the instant t, denoted by $\alpha(t)$, is an element of the finite set

$$A = \{\alpha_1, \alpha_2, ..., \alpha_r\} \tag{4}$$

- The transition probability matrix at any instant t, denoted by $P(t)$ which guides the choice of action at t

$$\begin{pmatrix} P_{11}, & \dots, & P_{1r} \\ \vdots & \vdots & \vdots \\ P_{s1}, & \dots, & P_{sr} \end{pmatrix} \tag{5}$$

where $P_{ij}(t)$ is the probability of choosing action *j* when the state at t is *i*.

- The input (which reflects the feedback from the environment) of an automaton at the instant t, denoted by $\beta(t)$, is either 0 or 1. 0 means the feedback is bad; 1 means the feedback is good.

- Ψ is an updating scheme, or a reinforcement learning algorithm, which generates $P(t+1)$ from $P(t)$.

Classical SLA reinforcement algorithms include General Linear-Reward-Penalty (LRP) Scheme, Symmetric Linear Reward-Penalty (SLRP) Scheme, Linear Reward-Inaction (LRI) Scheme and some non-linear schemes [8]. All these algorithms update the probability vector based directly on the environment feedback. Basically, when the input(environment feedback) is 1, these algorithms *reward* the chosen action by increasing its selection probability; the algorithm *penalize* it by decreasing its selection probability otherwise.

Kachroo *et. al.* [7] apply classical reinforcement learning algorithms and non-linear schemes[8] for DPM. The experiment results show that the SLA approach presents better results with respect to energy consumption and latency, compared to learning tree based approach [1], exponential-average approach [5] and OPBA [6]. However,

(1) All these classic learning algorithms update the probability matrix *prob* based directly on the environment feedback. Such algorithms may have high possibilities to converge to wrong decisions and before they learn the mistakes, the idle length distribution may change rapidly such that the new optimal decision is wrong again.

(2) The SLA design in [7] has high computational overhead. It is critical to keep the computation time of both probability updating and action selection low to support energy constrained real-time systems. In [7], for power state S_i ($i = 1, 2, \dots, k$) , there are $k-i+2$ allowable state transitions from S_i: S_i, S_{i+1}, S_{i+2}, ..., S_k and S_0. The allowable state transitions from S_0 include every state. So the size of the action set is $k+1+\sum_{i=1}^{k}(k-i+2) = 0.5(k^2+5k+2)$. The experiments in [7] show that the optimal learning algorithms are the non-linear learning algorithms, which requires more computation time than linear ones to update the probability for each action.

(3) The power state transition decisions are frequently made every $1ms$,$10ms$ or $100ms$. Frequent action selection may enhance the chances to make the system in optimal power states, but the power manager needs to compare the probability matrix with a random generated number for each action selection, which consumes possibly unnecessary computation time and power.

3 Proposed Algorithm

In this section, we propose a stochastic learning automaton to guide online DPM with low computational overhead. This approach differs from [7] in that: (1) the algorithm makes only one decision for each idle period, instead of making decisions at every timer tick, which avoid unnecessary decision makings; (2) the algorithm applies linear learning algorithms instead of non-linear ones, which

has relatively low computational overhead; (3) We further combine dynamic thresholds computed by OPBA with the stochastic learning automaton, which reduces the energy cost and latency effect introduced by wrong decision made by the automaton. This hybrid algorithm works as follows: Let *prob* be the vector that denotes the probability to transition the system from the active state to each low power state. When the system becomes idle, it generates a random number in the range of [0, 1]. By comparing the random number with *prob*, it makes a stochastic decision. The system will then transition to the power state according to the SLA decision. When the idle period exceeds the periodic updated OPBA thresholds for lower power states than the SLA decision, the hybrid algorithm works the same as OPBA. The SLA can eliminate the energy wasted before the threshold in the OPBA only approach, while the OPBA threshold may reduce the possible energy loss due to wrong decisions made by the SLA. We consider three hybrid algorithms: LRP-OPBA, AELA-OPBA, SELA-OPBA, where the learning algorithm for the SLA is LRP, AELA, SELA (presented in next section) respectively.

3.1 Stochastic Learning Algorithm

As we mentioned in the previous section, the critical part for the SLA design is to choose a learning algorithm that has fast convergence speed and adaptability to non-stationary workloads. In this section, we discuss our choice among the various classic reinforcement learning algorithms. We also apply two enhanced stochastic learning algorithms in our proposed algorithm.

3.1.1 Reinforcement Learning Algorithms

Classic learning algorithms include Linear Reward-Penalty (LRP), Linear Reward-Inaction (LRI) and some non-linear learning algorithms [8]. With respect to their Markovian representations, learning automata are classified into two main categories: ergodic automata and automata possessing absorbing barriers. When the reward probabilities of actions are time-variant (non-stationary environment), ergodic automata are preferred, because they are capable of being adapted to the environmental changes.

The LRI algorithm is not ergodic. It is possible to get stuck in absorbing states. This makes it sensitive to the starting conditions and probabilities, and also to non-stationary environments. Such occurs when action probabilities tend to one, so that if the environment changes, the probability vector may not adapt to the new optimum for a long time. Because we target for non-stationary workloads, we do not consider LRI as a candidate for learning algorithms. The non-linear learning algorithms are more complicated updating schemes than linear ones. Since the implementation must introduce little overhead for embedded systems, non-linear algorithms are not favorable.

In this paper, we apply three Linear Reward-Penalty (LRP) based learning algorithms to update the probability vector in SLA.

The LRP algorithm works as the follows: Let a_k be the last action that transitions the system from the active state to state k. Let b be the optimal action the offline algorithm would choose. Let α and β be the reward factor and penalty factor respectively, which are values in the range [0, 1]. Let nS be the number of power states. When a new service request arrives, we compute the offline decision b by simply comparing the last actual idle period length and the break-even time for each low power state. The automaton will evolve by updating the probability vector and adapt to the changing idle length distribution in the following way:

- If $b = a_k$, $prob[a_k] = prob[a_k] + \alpha(1 - prob[a_k])$; $prob[a_j] = prob[a_j] - \alpha * prob[a_j]$ $(j \neq k)$.
- if $b \neq a_k$, $prob[a_k] = prob[a_k] - \beta prob[a_k]$; $prob[a_j] = prob[a_j] + (\beta/(nS - 1) - \beta * prob[a_j])$ $(j \neq k)$.

The probability vector $p(n)$ in LRP has been shown to converge in distribution to a normal random variable for small step sizes. The scheme is also ergodic, so that this distribution function is independent of the initial probability vector $p(0)$. This feature is advantageous for non-stationary environments, as the automaton does not get stuck in absorbing states and so is better able to track the changing optimal probability vector.

3.1.2 Average Estimator Learning Algorithm

The Average Estimator Learning Algorithm (AELA) uses a running estimate of the mean reward by looking back into a history window and the environment feedback, rather than looking only on the environment feedback. Simulation results show that the average estimator learning algorithm has better performance than the above classic learning algorithms [10].

We associate W for each action to the SLA as the learning window size to record the reward history for each action.

After each idle period, we use LRP to update the most recent reward to each element in the probability vector. We then delete the oldest reward in the learning window of a_k and insert the reward/penalty by the Linear Reward-Penalty algorithm as the newest reward/penalty value for action a_k. We calculate the deterministic reward estimate D_k for a_k.

$$D_k = \frac{\sum_0^{w-1} r_i}{W} \tag{6}$$

where r_i (i = 0,1,...,$w-1$) are the rewards received during the last w times that action a_k was chosen.

In AELA-OPBA, we use this deterministic reward estimate to update the transition probabilities.

3.1.3 Stochastic Estimator Learning Algorithm

Papadimitriou [10] points out that the performance of AELA decreases when they operate in a non-stationary stochastic environment, because of the existence of old and consequently invalid, feedback information in the estimator. He further proposes the Stochastic Estimator Learning Algorithm (SELA) which varies the average deterministic reward by a small stochastic factor and shows improved performance. The simulation results [10] show that it achieves a high choice probability of the optimal action and high rate of adaptation to environmental changes in non-stationary environments.

We modify SELA for DPM. It works as the follows: we calculate the stochastic reward estimate before updating the transition probabilities. We associate each action an oldness number m_k, which keeps track of the time passed from the last time the action was chosen. We denote m_j as the oldness number of an opposite action of k. We update the oldness vector as in Line 4. The stochastic reward estimate S_k is then defined as:

$$S_k = D_k + N(0, \sigma^2) \tag{7}$$

where σ = min(βm_k, σ_{max})

$N(0, \sigma^2)$ is a random number selected with a normal probability distribution, with a mean of 0 and a variance of σ^2. β is an internal automaton's parameter that determines how rapidly the stochastic estimates become independent from the deterministic ones. σ_{max} is the maximum permitted value of σ. $N(0, \sigma^2)$ gradually makes the estimate deviate from the history records in a non-stationary stochastic environment. It may reduce the effect of those old and invalid records in the history window.

Algorithm 1 Stochastic Estimator Learning Algorithm

1: Compute the optimal offline action b;
2: Update the reward histogram based on the comparison of b and a_k;
3: Compute the new **deterministic estimate** of the mean reward of each action as is given by equation (6);
4: Update the **Oldness Vector** by setting m_k = 0 and m_j=m_j + 1 for all $j \neq k$;
5: For each action, compute the new **stochastic estimate** of mean reward u_s as is given by equation (7);
6: For each state, prob[s] = prob[s] + u_s;
7: Normalize the prob vector such that $\Sigma_{s=1}^{nS} prob[\mathrm{s}] = 1$

3.2 Implementation Consideration

In order to use online DPM algorithm in a practical embedded system, it must be implemented efficiently. OPBA and SLA based approach are the two best online algorithms for DPM in the literature in terms of energy savings and latency effects. To compute each threshold, OPBA has to search among n values to minimize (6). The time complexity of OPBA to update the thresholds is $O(kn)$ with an efficient implementation [6], where $k+1$ is the number of states and n is the number of bins in the histogram. The cost of implementation depends on the number of bins and the threshold update frequency. Note at each update time, OPBA has to find the best threshold for each power state by searching the best threshold among the n candidates to minimize the expected energy consumption equation defined in [6]. The time complexity of SLA based approach depends on the complexity of the learning algorithm and the number of actions used in the SLA as we described in Section 2. Our hybrid algorithm applies relatively simple SLA learning schemes rather than non-linear learning algorithms. Unlike [7], we only make decision when a new idle period appears and update probabilities when an idle period ends and do not make transitions at each time tick. In addition, we reduce the number of actions from $0.5(k^2+5k+2)$ to $k+1$. On the other hand, we may also reduce the update frequency and number of bins in the OPBA part, for the SLA helps to make decisions.

In order to evaluate the energy efficiency of our hybrid algorithms, simulation setup and results are described in the next section.

4 Experiments

We evaluate the SLA, OPBA and the hybrid algorithms with traditional prediction techniques by simulation. The algorithm test suite include:
(1) *Optimal Offline Algorithm (OPT)*: This algorithm is assumed to know the length of the idle period in advance and devises an optimal shutdown schedule accordingly. Note that the latency of OPT is 0, while other DPM algorithms may consume less energy at the cost of introducing higher latency.
(2) *DET*: The offline algorithm PLEA, which applies fixed thresholds for multiple power states. Details see [6].
(3) *Last Period (LAST)*: This simple algorithm uses the last period as a predictor for the next idle period.
(4) *Exponential Average(Exp-Avg)*: Proposed by Hwang *et. al.* [5], this policy uses a weighted sum of the last idle period T_{idle}^{n-1} and the last prediction T_{pred}^{n-1} to predict the new idle period T_{pred}^{n}:

$$T_{pred}^{n} = \lambda T_{idle}^{n-1} + (1-\lambda) T_{pred}^{n-1} \tag{8}$$

Table 1. IBM Mobile Hard-drive Power Model

State	$P(W)$	$E_{startup}(J)$	$L_{wakeup}(ms)$
Sleep	0	4.75	5000
Standby	0.2	1.575	1500
Idle	0.9	0.56	40
Active	1.9	0	0

Table 2. Competitive ratio and average latency

Algorithm	Competitive Ratio	Average Latency (ms)
OPT	1.000	0
DET	1.429	917.816
OPBA	0.991	1660.372
Hybrid	0.937	1401.588
LAST	1.754	1737.017
LAST-P	1.009	1054.688
EXP	0.996	1560.543
EXP-P	1.139	1135.512
TREE	1.003	1966.419
TREE-P	2.156	1313.641
LRP	0.964	1632.380
LRP-P	2.611	744.617
AELA	0.962	1659.522
AELA-P	2.423	807.788
SELA	1.031	2280.945
SELA-P	2.276	1841.996

where λ is a value in the range of (0,1).
(5) *Adaptive Learning Tree (Tree)*[1]: the approach uses an adaptive learning tree to predict the power state for the next idle period based on the recent history of idle period lengths.
(6) *Linear Reward-Penalty (LRP)*: SLA approach using LRP algorithm.
(7) *Average Estimator Learning Algorithm (AELA)*: SLA approach using AELA algorithm.
(8) *Stochastic Estimator Learning Algorithm (SELA)*: SLA approach using SELA algorithm.
(9) *Online Probability-Based Algorithm (OPBA)*[6].
(10) *LRP-OPBA*.
(12) *AELA-OPBA*.
(13) *SELA-OPBA*.

In addition, for Last, Exp-Avg, Tree, LRP, AELA, SELA, we consider two versions of each algorithm as in [6, 7]: *preemptive-wake-up* and *on-demand-wake-up*, respectively. (Last-P, Exp-Avg-P, Tree-P, LRP-P, AELA-P, SELA-P are the preemptive-wake-up versions.) The *preemptive-wake-up* versions consider reducing the latency introduced by the DPM algorithms by predictively waking up the system before a new service request arrives.

All history based algorithms have the same window size

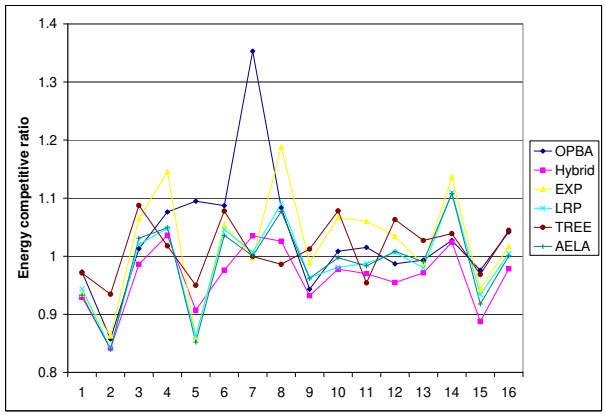

Figure 1. Energy comparison for each trace on OPBA, Hybrid, EXP, TREE, LRP, AELA

of 100 for learning. The OPBA threshold update frequency is set to 10. We use the concept of competitive ratio as the normalized comparison (normalized by the offline algorithm) for energy savings. We also compare the latency effects for each DPM algorithm.

To compare the competitive ratios for energy savings and latency effects, we use the mobile hard-drive power model from IBM and the disk traces collected from auspex file server archive. The mobile hard-drive has four power states as illustrated in Table 1, where columns P, $E_{startup}$, L_{wakeup} represent the power consumption of each power state, the startup energy and the wakeup latency from each low power state to the Active state. Note that [6, 3, 7] use the same data and power model to evaluate DPM algorithms.

The auspex file server archive include 16 traces with different lengths. We report the average of the results for each individual trace weighted by length. Table 2 shows the competitive ratio and average latency (ms) for each algorithm. For the three hybrid algorithms, the experiment results are very similar for LRP-OPBA and AELA-OPBA, but AELA-OPBA has 6% less average latency. The performance of SELA-OPBA has a large variation across different traces and has worse average energy savings than the other two. We only include the results for AELA-OPBA (named as Hybrid) in the figures and the corresponding discussions.

The experiment results show that the hybrid algorithm is the best algorithm in terms of both energy and latency. It has a consistent better energy efficiency than OPBA with 15.6% lower latency and has on average of 5% and up to 20% more energy savings (Table 2 and Figure 1) than OPBA for the auspex hard disk traces. As compared to the SLA algo-

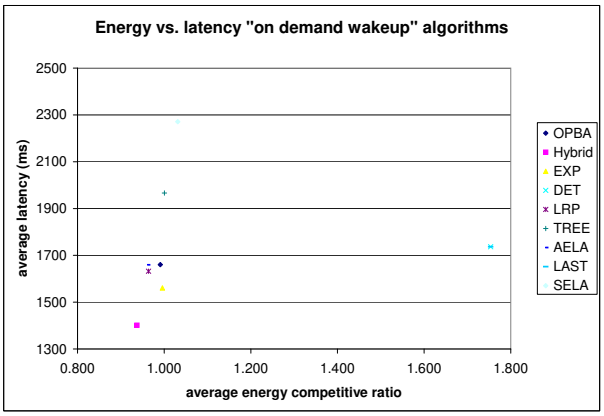

Figure 2. Comparison for "on-demand wakeup" algorithms

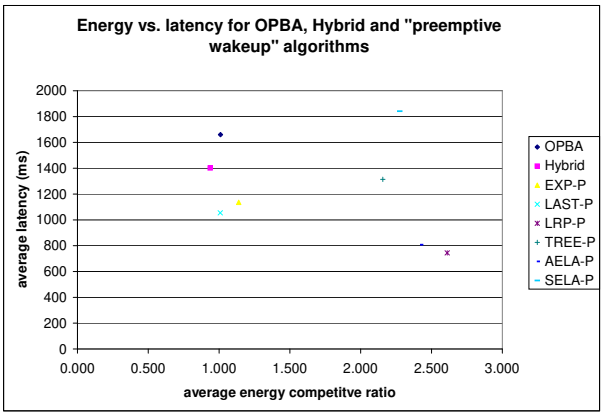

Figure 3. Comparison for OPBA, Hybrid & "preemptive wakeup" algorithms

rithms (LRP and AELA) and EXP that have similar energy savings, the hybrid algorithm reduce the latency on the average of 14%.

Figure 2 and Figure 3 compare the average energy competitive ratio vs. average latency of OPBA and the hybrid algorithm with each"on-demand wakeup" algorithm and each "preemptive wakeup" algorithm respectively.

The hybrid algorithm has the advantage of Last, Exp-Avg, Tree and SLA (LRP, AELA, SELA) to avoid energy wasted before reaching the thresholds and the advantage of threshold based algorithms (Timeout, DET, OPBA) to have low misprediction rate and low average latency. System level dynamic power management is an engineering design decision which should be tailored to meet the energy and latency requirements and minimize implementation overhead. The learning algorithms and the hybrid algorithms enrich the design space for actual engineering optimization between energy and latency constraints.

5 Conclusions

In this paper, we propose a hybrid algorithm which combines the online probability based algorithm and the stochastic learning automata based algorithm to improve online dynamic power management for battery-operated embedded systems. The choice of learning algorithms relies on how the recent history of idle period lengths can be used to predict the upcoming idle period length. LRP and AELA give the same weight to each entry in the histogram, while SELA adds a stochastic noise factor to deviate the information retrieved from the recent history. Our hybrid algorithm enhances the OPBA algorithm by directly transitioning the system to the state predicted by an SLA, so that it can avoid the energy spent before the OPBA reaches the optimal state. Both OPBA and SLA based algorithms reduce the latency effects than other online DPM algorithms. The hybrid algorithm has a consistent on an average of 5% and up to 20% more energy savings than OPBA, while introduce on the average of 14% lower latency than any other algorithm that has similar energy savings.

References

[1] E. Chung, L. Benini, and G. DeMicheli. Dynamic power management using adaptive learning tree. In *Proceedings of the 1999 IEEE/ACM International Conference on Computer-Aided Design*, 1999.

[2] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. D. Micheli. Dynamic power management for nonstationary service requests. *IEEE Transactions on Computers*, 2002.

[3] T. Erbes, S. Shukla, and P. Kachroo. Stochastic learning feedback hybrid automata for power management in embedded systems. In *Technical Report No:2004-03, FERMAT Group, Virninia Polytechnic Institute and State University*, 2004.

[4] R. K. Gupta, S. Irani, and S. K. Shukla. Formal methods for dynamic power management. In *Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design*, 2003.

[5] C. Hwang and A. Wu. A predictive system shutdown method for energy saving of event-driven computation. *ACM Transactions on Design Automation of Electronic Systems*, 2000.

[6] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Transactions on Embedded Computing Systems*, 2003.

[7] P. Kachroo, S. Shukla, T. Erbes, and H. Patel. Stochastic learning feedback hybrid automata for power management in embedded systems. In *IEEE International Workshop on Soft Computing in Industrial Applications*, 2003.

[8] K. Narendra and M. Thathachar. *Learning Automata: An Introduction*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[9] G. A. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli. Policy optimization for dynamic power management. In *Proceedings of the 35th Design Automation Conference*, 1998.

[10] G. Papadimitriou. A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 1994.

[11] Q. Qiu, Q. Wu, and M. Pedram. Stochastic modeling of a power-managed system: construction and optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2001.

[12] T. Simunic, L. Beniani, and G. D. Micheli. Event-driven power management of portable systems. In *Proceedins of the 12th International Symposium on System Synthesis*, 1999.