

A Power-Aware, Best-Effort Real-Time Task Scheduling Algorithm

Jinggang Wang, Binoy Ravindran, and Tom Martin
The Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061, USA
E-mail: {jiwang5,binoy,tlmartin}@vt.edu

Abstract

In this paper, we present a power-aware, best-effort real-time task scheduling algorithm called PA-BTA that optimizes real-time performance and power consumption. The algorithm considers a timeliness model where task timing constraints are described using Jensen's benefit functions and a system-level power model. We propose a metric called "Energy and Real-Time Performance Grade" (ERG) to measure real-time performance and power consumption in a unified way. Since the scheduling problem is NP-hard, PA-BTA heuristically computes schedules to maximize ERG, incurring a worst-case computational cost of $O(n^2)$. Our simulation results indicate that the algorithm performs close to the optimal algorithm and better than other algorithms considered in the study.

1. Introduction

Power consumption is increasingly becoming a major design constraint in modern computing systems ranging from mobile and embedded systems to desktop computers and network routers. Broadly considered, power management has been addressed in terms of two different but related concerns. These include low power and power-aware.

Research on low power computing has been pursued at multiple system levels over the last decade. These levels include, from bottom to top, VLSI design-level, architecture-level, operating system-level, and compiler/application-level [8] [4] [10].

However, recent research in low-power design have considered systems that have varying power availabilities [2, 5]. Such systems need to dynamically adapt to power availability, performance, and requirements such as reliability. For example, when the power availability is good, the system should favor considerations of performance or reliability more than those of power.

Thus, researchers have proposed the power-aware design objective that seeks to make the best use of the available power. Furthermore, the design objective seeks to monitor the power availability and adapt to low-power or high performance [2]. Thus, a low-power system can be considered as a special case of a power-aware system.

Research in power-aware computing has also been pursued at multiple system levels. In this paper, we advance the power-aware computing research at the operating system-level by focusing on power-aware real-time scheduling. We present a new power-aware, real-time scheduling algorithm called PA-BTA. Though power-aware real-time scheduling has been studied in the past [2, 5], our work is novel in two fundamental aspects: (1) we consider timing constraints that are described using *Jensen's benefit functions* and (2) we consider a *system-level* power model.

1.1. Soft Timing Constraints and Soft Timeliness Optimality

Most of the past efforts on power-aware real-time scheduling focus on deadline timing constraints and the hard timeliness optimality criterion that "all deadlines must be satisfied."

Deadline timing constraints are "hard" in the sense that completing a deadline-constrained activity before its deadline implies the accrual of some "benefit" and that benefit remains the same if the activity were to complete anytime before the deadline. Furthermore, completing the activity after the deadline implies a timing failure i.e., the accrual of zero benefit.

With deadlines, it therefore becomes difficult to express timing constraints that are not hard, but "soft" in the sense that completing a time-constrained activity at anytime will result in some benefit and that benefit *varies* with activity completion time. Furthermore, with deadlines, it becomes difficult to specify a collective timeliness optimality criterion that is not hard, but soft in the sense that completing as many soft time-

constrained activities as possible at their optimal completion times—completion times that will lead to maximal benefit—is what is desirable.

Furthermore, to achieve the hard timeliness optimality criterion, the hard real-time theory [11] makes extremely strong presumptions about application behavior such as deterministic postulations on application workloads — e.g., upper bounds on activity arrival rates and execution times — that are assumed to be always respected at run-time. This timeliness optimality is effective for low-level real-time systems that are used to manage controlled, predictable physical systems such as that for unit-level, periodic monitoring and regulatory control.

However, the hard timeliness optimality criterion is ineffective for supervisory, real-time, embedded control systems such as those that are emerging in defense, industrial automation, and telecommunication domains [6, 7]. Such systems control large, physical systems, and are fundamentally distinguished by the significant run-time uncertainties that are inherent in their application environment. Consequently, upper bounds on application workloads are not known to exist at design time with sufficient accuracy. Hence, achieving the hard timeliness optimality is not cost effective for such systems.

Jensen's benefit functions [6] precisely address the specification of soft time constraints. Furthermore, Jensen's benefit accrual model [6] allows the specification of soft timeliness optimality criteria.

1.2. System-Level Power Model

Most of the past efforts on power-aware real-time scheduling consider a power model that only takes into account the dynamic power consumption of the CPU. However, the battery life of a system is determined by the *system's* energy consumption, besides the energy consumption of the CPU. Therefore, power models that are used in past efforts are not accurate for prolonging the battery life.

Based on the experimental observations that some components in computer systems consume constant power and some consume power that is only scalable to either voltage or frequency, the third authors' prior work has resulted in a new power model for measuring power consumption [12]. In this model, power consumption is measured as $P = C_3 \times f^3 + C_2 \times f^2 + C_1 \times f + C_0$, where f is the processor clock frequency and C_0 , C_1 , C_2 , and C_3 are system parameters [13]. In Section 2.2, we further elaborate on this power model.

1.3. Objectives, Overview, and Road-Map

To integrate soft timeliness optimality and energy consumption into a single system-level performance metric, we propose the concept of “Energy and Real-time performance Grade”(or ERG). ERG essentially allows us to optimize soft timeliness and energy consumption in a unified way.

Given the system metric of ERG, our objective is thus to design a real-time scheduling algorithm that computes real-time schedules such that ERG is maximized. This scheduling problem can be shown to be NP-hard. Thus, we present a heuristic algorithm for this problem called Power-Aware, Best-Effort, Real-Time Task Scheduling Algorithm (or PA-BTA) that heuristically computes schedules to maximize ERG.

The PA-BTA algorithm is “best-effort” in the sense that it seeks to provide the best ERG to the application, where the best ERG that the application can accrue is application-specified (using Jensen's benefit functions and the ERG metric). The algorithm builds upon our prior work on best-effort real-time packet scheduling, which had resulted in the BPA algorithm [14].

We evaluate the performance of PA-BTA through simulation. Our simulation studies reveal that PA-BTA performs very close to the optimal algorithm. Furthermore, we observe that PA-BTA significantly outperforms real-time scheduling algorithms including BPA [14], DASA [3], and EDF [11] for most situations.

Thus, the contributions of the paper are twofold: (1) a metric called ERG, for optimizing soft timeliness and system-level power consumption in a unified manner; and (2) the PA-BTA task scheduling algorithm that optimizes ERG.

The rest of the paper is organized as follows: We discuss the timeliness model, power model, and the ERG metric in Section 2. We formally describe the problem and state our objectives in Section 3. In Section 4, we describe the PA-BTA algorithm, the heuristics used by the algorithm, and the rationale behind the heuristics. In Section 5, we evaluate the performance of PA-BTA and report our results. Finally, the paper concludes with a summary of the work and discusses future efforts in Section 6.

2. Timeliness Model, Power Model, and ERG Metric

2.1. Timeliness Model

We consider a set of tasks that execute in a uni-processor system. We denote the set of tasks as $t_i \in A, i \in [1, n]$. All tasks are assumed to be

independent i.e., they do not share resources or have any precedence relationships. Furthermore, they are assumed to be non-preemptive.

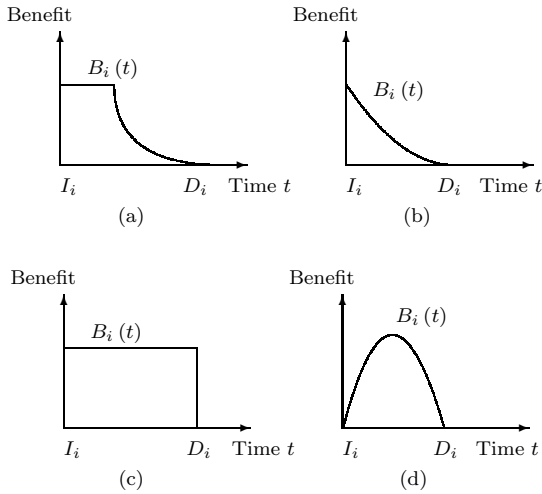


Figure 1. Example Unimodal Functions

We use Jensen’s benefit functions [6] for expressing application timing requirements. The benefit function of a task $t_i \in A$ is denoted as B_i . Thus, the completion of a task t_i (since the release of the task) at a time t will yield a benefit $B_i(t)$.

Though benefit functions can take arbitrary shapes, in this paper, we focus on *unimodal* benefit functions that are *non-increasing*. Unimodal benefit functions are those benefit functions for which any decrease in benefit cannot be followed by an increase in benefit [6]. Benefit functions, which are not unimodal are called *multimodal*.

Example unimodal functions are shown in Figure 1. Note that the classical “hard” deadline can be expressed as a “rectangular” unimodal function such as the one shown in Figure 1(c), where the completion of a task at anytime before its deadline will result in uniform benefit; the completion of the task after the deadline will result in zero benefit. All non-rectangular benefit functions express soft timing requirements [6].

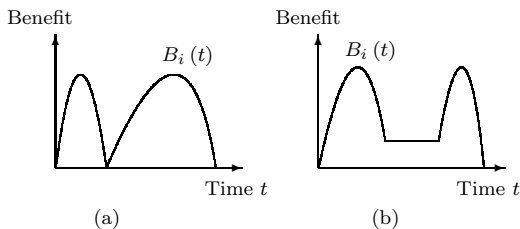


Figure 2. Example Multimodal Functions

Example multimodal benefit functions are shown in Figure 2.

Unimodal that are non-increasing are simply those benefit functions for which benefit never increases when time advances. Figures 1(a), 1(b), and 1(c) show examples. The class of non-increasing unimodal functions allow the specification of a broad range of timing constraints, including hard constraints and a majority of soft constraints. Therefore, we focus on non-increasing unimodal benefit functions here.

All benefit functions $B_i, i \in [1, n]$ have an initial time I_i and a deadline time D_i . Initial time is the earliest time for which the function is defined and deadline time is the time at which the function drops to a zero value. Furthermore, $B_i(t) \geq 0, \forall t \in [I_i, D_i], i \in [1, n]$ and $B_i(t) = 0, \forall t \notin [I_i, D_i], i \in [1, n]$.

2.2. Energy Consumption Model

We assume that the processor can execute tasks at m distinct levels of clock frequency. When the processor operates at a clock frequency f , the dynamic power consumption of the processor, denoted as P_d , is given by $P_d = C_{ef} \times V_{dd}^2 \times f$, where C_{ef} is the effective switch capacitance and V_{dd} is the supply voltage.

On the other hand, the clock frequency is almost linearly related to the supply voltage, since $f = k \times \frac{(V_{dd} - V_t)^2}{V_{dd}}$, where k is constant and V_t is the threshold voltage [4]. By approximation, we assume $f = a \times V_{dd}$, where a is constant. Thus, we obtain $P_d = \frac{C_{ef}}{a^2} \times f^3$, which is equivalent to $P_d = C_3 \times f^3$, where C_3 is constant. In this case, both the supply voltage and the clock frequency can be scaled.

In addition to the processor, there are also other components in a computer system that consume power. Given the dynamic power consumption equation $P_d = C_{ef} \times V_{dd}^2 \times f$, we can derive power consumption equations for all other system components.

Some components in the system must operate at a fixed voltage and thus their power can only scale with frequency. Examples include main memory. In this case, $C_{ef} \times V_{dd}^2$ can be represented as another constant such as C_1 , and the equation becomes $P_d = C_1 \times f$.

Other components in the system consume constant power with respect to the clock frequency. Examples include display devices and speakers. Thus, we can represent their power consumption as C_0 , where C_0 is constant.

Finally, for completeness in fitting the measured power of a system to the cubic equation, we include another term to represent the quadratic term i.e., $P_d = C_2 \times V_{dd}^2$. Since f is almost

linearly related to V_{dd} , we can represent P_d as $P_d = C_2 \times f^2$. While this term does not represent the dynamic power consumption of CMOS, because it implies that V_{dd} is being lowered without also lowering f , in practice, this term may appear because of variations in DC-DC regulator efficiency across the range of output power, CMOS leakage currents, and other second order effects [12].

Summing the power consumption of all system components together, we obtain a single equation for the system-level power consumption for each task as $P = C_3 \times f^3 + C_2 \times f^2 + C_1 \times f + C_0$.

The corresponding energy consumption is given by $E = P \times T_i$, where T_i is the execution time of task t_i .

The task execution time is also related to the processor clock frequency. The first order relationship between the task execution time T_i and the clock frequency f can be represented as $T_i = k/f$, where k is constant. This implies that task execution time is inversely proportional to clock frequency [15].

Based on the above two expressions, the energy consumption of each task can be represented as $E = k \times C_3 \times f^2 + k \times C_2 \times f + k \times C_1 + k \times \frac{C_0}{f}$. This equation indicates that there is an optimal value for clock frequency that minimizes E for each task.

2.3. The ERG Metric

Given the timeliness model, when a real-time task completes, it contributes some timeliness benefit as specified by the task benefit function. Furthermore, the task also consumes some power. To integrate real-time performance and power consumption into a single performance metric, we propose the metric *Energy and Real-time performance Grade* (or ERG). ERG is defined as $ERG = \frac{BF_r}{BF_{\max}} \times BW + \frac{E_{\max} - E_r}{E_{\max}} \times PW$, where BF_r is the actual aggregate timeliness benefit that is accrued, BF_{\max} is the maximum possible aggregate timeliness benefit that can be accrued, E_{\max} is the maximum energy that can be consumed, E_r is the actual energy consumed, BW is the weight of real-time performance, and PW is the weight of energy consumption.

The ERG metric can easily adapt to multiple design objectives. For example, if system power is not a concern and we simply want to obtain the best real-time performance, the weight of energy consumption PW can be set to 0. On the other hand, if system power is more important than real-time performance, then the weight of energy consumption PW can be set to be larger than the weight of real-time performance BW . In cases where real-time performance and power consumption are equally important, BW and PW can be

set to be equal.

3. Problem Definition and Objectives

The total system ERG is obtained by simply summing the ERG of each individual task. Therefore, from the definition of ERG, we can see that larger the ERG, larger is the aggregate real-time benefit and smaller is the system-level power consumption. Therefore, the system design objective can be defined as maximizing ERG i.e., maximizing $\sum_{k=1}^n ERG_K(T_k)$, where $ERG_K(T_k)$ denotes the ERG obtained from task t_k when t_k completes at time T_k .

The objective of the PA-BTA algorithm can therefore be formalized as follows: Let $\mathcal{A} \subseteq A$ denote the set of tasks in the processor task queue at time T . Let α denote the number of tasks in the set \mathcal{A} . Let $S(\mathcal{A})$ denote all possible sequences of tasks of the set \mathcal{A} and let $\sigma \subseteq S(\mathcal{A})$ denote one of the possible task sequence of the tasks in \mathcal{A} . Let $\sigma(i)$ denote the task occupying the i^{th} position in the schedule σ . Then, the objective of PA-BTA is to: *Maximize* $ERG(\sigma) = \sum_{K=1}^{\sigma} ERG_{\sigma(K)}(T + T_k)$, where $T_k = \sum_{i=1}^k T_{\sigma(i)}$ and $\sigma \subseteq S(\mathcal{A})$.

Thus, informally, the objective of the PA-BTA algorithm is to determine a task schedule and the associated processor frequency level for each task such that the schedule will maximize the aggregate task ERG (or the sum of the individual task ERGs).

This scheduling problem is equivalent to the NP-hard scheduling problem considered in [14, 1], excluding the power optimization constraint. Thus, the scheduling problem with the power optimization constraint is NP-hard.

4. The PA-BTA Algorithm: Heuristics and Rationale

4.1. Sort Tasks in Decreasing Order of “Return of Investments”

The potential ERG that can be obtained by spending a unit amount of processor time for a task defines a measure of the “return of investment” for the task. Thus, by ordering tasks in the decreasing order of their return of investments, we can “greedily” collect as much “high return” tasks into the schedule as early as possible. This is because a task included in the schedule at any instant is always the one with the next “highest-return” value among all non-examined tasks. This will increase the likelihood of maximizing the aggregate ERG.

We can approximate the return of investment for a task as the ratio of the maximum possible ERG that can be obtained for the task to the task relative deadline. This is just a single division costing $O(1)$ time. We determine the maximum possible ERG for a task using the optimal processor frequency level for the task.

4.2. Drop Infeasible Tasks From Schedule

Infeasible tasks are tasks that cannot complete before their deadlines, no matter what. This is because, the remaining execution time of such tasks are longer than the time interval between the scheduling instant—the time at which the scheduler is triggered, which is the arrival or departure of a task—and the task deadlines. Tasks that are not infeasible are feasible tasks.

By dropping infeasible tasks from the schedule, we collect as much feasible tasks into the schedule as possible. Furthermore, such feasible tasks are inserted into the schedule-beginning, as tasks are examined in decreasing order of their return of investments. This will increase the likelihood of maximizing the aggregate task ERG, as feasible tasks yield greater benefit and thus greater ERG if they complete earlier, since we are focusing on non-increasing unimodal functions.

Furthermore, infeasible tasks yield zero timeliness benefit and thus close to zero ERG, if they complete after their deadlines. Thus, there is no reason for executing them and jeopardize the potential benefit and the ERG that can be accrued from feasible tasks.

4.3. Maximize Local ERG As Much As Possible

We derive the notion of *local aggregate ERG* from the *precedence-relation* property presented in [1].

Consider two schedules $\sigma_a = \langle \sigma_1, t_i, t_j, \sigma_2 \rangle$ and $\sigma_b = \langle \sigma_1, t_j, t_i, \sigma_2 \rangle$ of a task set \mathcal{A} , such that $\sigma_1 \neq \emptyset$, $\sigma_2 \neq \emptyset$, $\sigma_1 \cup \sigma_2 = \mathcal{A} - \{t_i, t_j\}$, and $\sigma_1 \cap \sigma_2 = \emptyset$. Consider a (scheduling) time instant $t = \sum_{k \in \sigma_1} T_k$, when a scheduling decision has to be made. That is, t is the instant in time after all tasks in the schedule σ_1 has been completed.

Now, the scheduling decision at time t can be made by determining $\Delta_{i,j}(t)$, where: $\Delta_{i,j}(t) = [ERG_i(t + T_i) + ERG_j(t + T_i + T_j)] - [ERG_j(t + T_j) + ERG_i(t + T_j + T_i)]$. Thus, if $\Delta_{i,j}(t) \geq 0$, then schedule σ_a will yield a higher aggregate ERG than schedule σ_b . Otherwise, σ_b is better than σ_a .

Now, by examining adjacent tasks t_i and t_j in a schedule $\langle \sigma_1, t_i, t_j, \sigma_2 \rangle$ and ensuring that $\Delta_{i,j}(t) \geq 0$, we can maximize the *local* aggregate ERG of

tasks t_i and t_j . If all adjacent tasks in the schedule have such locally maximized aggregate ERG, this will increase the likelihood of maximizing the global aggregate ERG.

Thus, PA-BTA exhaustively searches all the frequency levels of adjacent tasks t_i and t_j . The algorithm then determines the scheduling order for the two tasks and the associated best frequency level for each task such that the local aggregate ERG is maximized.

```

PA-BTA( $\mathcal{A}$ ,  $\alpha$ ,  $t$ )
/*  $\mathcal{A}$ : task set;  $\alpha$ : # of tasks in  $\mathcal{A}$ ;
 $t$ : time of scheduling event */
1.  $\sigma = \emptyset$ ; /* Initialize task schedule to empty; */
2. For each task  $t_i \in \mathcal{A}$ 
   2.1 Determine the optimal frequency  $f$ ;
   2.1 ReturnOfInvest =  $ERG(f)/D_i$ ;
3. Sort tasks in  $\mathcal{A}$  in decreasing order of
   their ReturnOfInvest; /*  $\mathcal{A}$  is now sorted */
4.  $\sigma = \mathcal{A}$ ; /*  $\sigma$  is set equal to sorted set  $\mathcal{A}$  */
5. For  $k = 1$  to  $\alpha$ 
   5.1 InOrder = TRUE;
   5.2 For  $i = 1$  to  $\alpha - 1$ 
     (1)  $j = i + 1$ ; /*  $t_j$ : task following  $t_i$  in  $\sigma$  */
     (2) For all frequency level of  $t_i$  and  $t_j$ 
         /* implement in 2 loops */
         • If ( $t_i$  is not feasible)
             ◇ Mark  $t_i$  as not feasible;
         • If ( $t_j$  is not feasible)
             ◇ Mark  $t_j$  as not feasible;
         • Compute  $ERG(i, j)$  and  $ERG(j, i)$ ;
         • Select the largest ERG within all
           combinations of frequency;
     (3) If largest ERG requires swap of  $t_i$  and  $t_j$ 
         • Swap( $t_i, t_j$ );
         •  $T = T + T(t_j)$ ; /*  $T(t_j)$ : execution
           time of  $t_j$  in current  $f$  */
         • InOrder = FALSE;
     (4) Else
         •  $T = T + T(t_i)$ ; /*  $T(t_i)$ : execution
           time of  $t_i$  in current  $f$  */
   5.3 If (InOrder = TRUE) Break;
   /* No swaps; so all tasks are inorder */
6.  $\sigma$  is final schedule; return task  $\sigma(1)$ ;

```

Figure 3. Pseudo-Code of PA-BTA

The maximization of the local aggregate ERG can be done in a manner similar to that of Bubble sort. We can examine adjacent pairs of tasks in the schedule, compute Δ , and swap the tasks, if the reverse order can lead to higher local aggregate ERG. Furthermore, the procedure can be repeated until no swaps are required.

Thus, the PA-BTA algorithm computes task schedules according to the heuristics discussed in Sections 4.1, 4.2, and 4.3. Pseudo-code of the algorithm at a high-level of abstraction is shown in Figure 3.

4.4. Computational Complexity of PA-BTA

The computational cost of PA-BTA depends upon the complexity of Step (5). The complexity of all other steps are dominated by this step.

The complexity of Step (5) is dominated by that of Step (5.2); all other sub-steps of Step (5) take $O(1)$ time. Step (5.2) can iterate a maximum of α times, and therefore costs $O(\alpha)$. Step (5) can iterate a maximum of α times, and thus costs $O(\alpha^2)$. Given n tasks, the complexity of PA-BTA is therefore $O(n^2)$.

5. Performance Evaluation

To evaluate the performance of PA-BTA, we consider two classes of experiments. In our first class of experiments, we compare PA-BTA with the optimal algorithm. The optimal algorithm exhaustively searches all possible schedules and all possible frequency levels for each task in a schedule to determine the optimal schedule. Since the resulting computational cost is large, we consider a small number of tasks and a small range of frequency levels for this class of experiments.

In our second class of experiments, we compare PA-BTA with other real-time scheduling algorithms including BPA [14], DASA [3], and EDF [9]. We consider a large number of tasks and a large range of frequency levels in this study.

Our motivation to consider BPA is that, in [14], BPA has been shown to be the best algorithm for the exact same task model and timeliness model considered by PA-BTA, except that it is not power-aware. Furthermore, we consider DASA and EDF because DASA outperforms EDF during overload situations and performs the same as EDF during under-load situations where EDF is optimal [3, 9]. Though DASA and EDF only allow deadlines as timing constraints, nevertheless, deadlines are a special case of the timeliness model considered by PA-BTA.

Since BPA, DASA, and EDF are not power-aware and do not select frequency levels for each individual task, we consider three versions for each of the algorithms that use three distinct frequency levels for the tasks. The levels include the highest, the medium, and the lowest frequency. We refer to the algorithm versions as BPA_h, BPA_m, BPA_l, DASA_h, DASA_m, DASA_l, EDF_h, EDF_m, and EDF_l, where “h,” “m,” and “l” imply the highest, medium, and the lowest frequency level, respectively.

We now describe the experimental environment and the two classes of experiments in the subsections that follow:

5.1. Experimental Environment

5.1.1 Parameters of Timeliness Model

To study the performance of the algorithms in a large data space, we randomly generate all task parameters including execution times, deadlines, and maximum benefit of the benefit functions using probability distribution functions.

We consider six benefit functions in our study. Three of the functions are shown in Figure 1. These include “soft-rectangular,” “exponential,” and “rectangular,” and are shown in Figures 1(a), 1(b), and 1(c), respectively. The other three functions include “linear,” “quadratic,” and “composite” functions. These are not shown here due to space limitations.

With different benefit functions, it is possible to have task models that are homogeneous and heterogeneous with respect to benefit function types. In the homogeneous case, all tasks have the same type of benefit function, while in the heterogeneous case, tasks can have different benefit function types. We consider both cases in our study.

5.1.2 Parameters of Power Model

To use the power model discussed previously, we select typical values based on system examples for parameters including C_3 , C_2 , C_1 , C_0 , and the base frequency. The base frequency is the minimum frequency the processor can work on, and all high frequencies are multiples of it.

In practice, the C_3 , C_2 , C_1 , and C_0 terms depend upon the power management state of the system and its subsystems. For example, if a laptop has its display on, the C_0 term will be large relative to the others. But if the display has been turned off, the C_0 term will be much smaller.

Different types of systems will also have different relative values for the C terms: The C_3 term is probably a much larger fraction of the total power in a PDA such as an iPAQ than it is in a notebook computer.

5.1.3 Real-Time versus Power: 3 Cases

Since the ERG metric allows relative importance to be defined for real-time performance and power consumption, we consider three cases. These include: case (1), where $PW = 2BW$ i.e., power consumption is more important than real-time performance; case (2), where $BW = 2PW$ i.e., real-time performance is more important than power consumption; and case (3), where $BW = PW$ i.e., real-time performance and power consumption are equally important.

5.2. Performance Results for Small Task Set and Frequency Range

For this class of experiments, we considered six tasks that can operate at five frequency levels. Figure 4 shows the average of the normalized aggregate ERG for all algorithms for case 1 (i.e., $PW = 2BW$) and under small task set and frequency range. The normalized aggregate ERG of an algorithm is simply the ratio of the aggregate ERG produced by the algorithm to that produced by the optimal algorithm. Since the optimal algorithm has a normalized aggregate ERG of 1, we do not show this in the figure. The measurements were obtained from hundreds of repeated experiments.

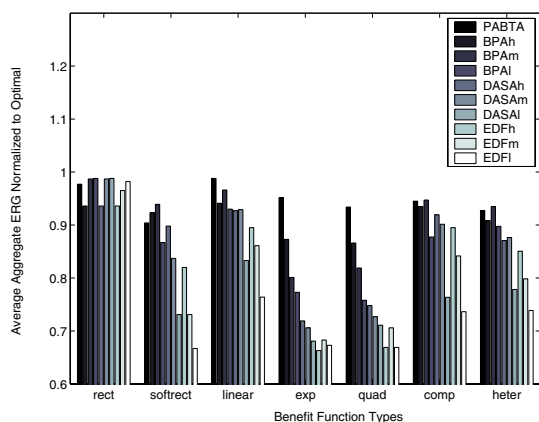


Figure 4. Performance of Algorithms When $PW = 2BW$

From Figure 4, we observe that the performance of PA-BTA is very close ($\geq 90\%$) to that of the optimal algorithm for homogenous benefit function types (for all six functions) as well as for heterogenous functions. Further, we observe that PA-BTA performs the best for all benefit functions, except rectangular benefit function. Furthermore, we observe that PA-BTA performs the best under linear benefit function. We believe that this is because the “return of investment” estimated by the algorithm under the linear function matches the real “return of investment” i.e., the ratio of the maximum benefit to the deadline is the real slope of the linear function.

We also observe that DASA performs the best among all algorithms other than PA-BTA, under rectangular benefit function. This further demonstrates the effectiveness of DASA for deadline constraints [3]. However, DASA performs poorly under other functions as it is not designed for them.

BPA is very close to DASA under rectangular function and much better than DASA under all other functions. This is not surprising as BPA

is precisely designed for non-increasing unimodal functions. Note that EDF performs the worst here.

Figure 5 shows the results for case 2 (i.e., $BW = 2PW$) and under small task set and frequency range. When compared with Figure 4, we observe that PA-BTA performs very good for linear, exponential, and quadratic benefit functions. However, it performs worse under rectangular, soft-rectangular, composite, and heterogeneous cases.

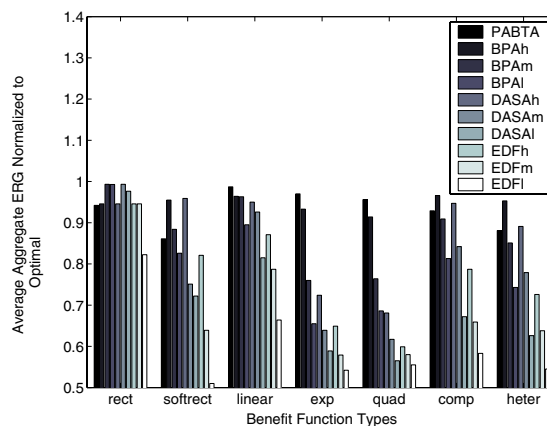


Figure 5. Performance of Algorithms When $BW = 2PW$

The figure also shows that BPA performs quite well and even better than PA-BTA under some frequency level and benefit functions. This indicates that the advantage of PA-BTA is more significant when power consumption is more important than real-time performance.

For the case when real-time performance and power consumption are equally important, we obtained results that furthered the aforementioned trend. We observed that PA-BTA performs better in case 3 (i.e., $BW = PW$) than it does in case 2 (i.e., $BW = 2PW$) and worse than it does in case 1 (i.e., $PW = 2BW$). This strengthens the result that when power consumption is more important, the advantage of PA-BTA is more significant. Due to space limitations, we do not show these results here.

5.3. Performance Results for Large Task Set and Frequency Range

For this class of experiments, we considered fifty tasks and thirty frequency levels. Since the optimal algorithm is not practical for this class, we normalized the final results of all algorithms with respect to that of PA-BTA.

Due to space limitations, we do not show results of this class of experiments here. The re-

sults for all three cases — (1) $PW = 2BW$, (2) $BW = 2PW$, and (3) $BW = PW$ — indicated that PA-BTA yields the highest aggregate ERG for all benefit functions, except for rectangular and soft-rectangular functions. For rectangular and soft-rectangular functions, we observed that BPA and DASA outperform PA-BTA only for some frequency levels.

Since the better performance of BPA and DASA (for the two benefit functions) were frequency-dependent, we considered an “average version” of BPA and DASA that used the average of the highest, medium, and lowest frequency levels for the tasks. We compared the average versions with PA-BTA and observed that PA-BTA performed better.

Our experiments in this class for the three cases also confirmed our earlier result that when power consumption was more important, the advantage of PA-BTA was more significant.

6. Conclusions, Future Work

In this paper, we present a power-aware, best-effort real-time scheduling algorithm called PA-BTA that optimizes real-time performance and power consumption in a unified manner. The novel aspects of the algorithm are that it considers flexible timing constraints that are described using Jensen’s benefit functions and a system-level power model. Our simulation results indicate that PA-BTA performs very close to the optimal algorithm and outperforms other algorithms for most cases. Furthermore, we observe that when power consumption is considered to be more important than real-time performance, the advantage of the algorithm is more pronounced.

Several aspects of PA-BTA are currently being studied. These include extending the algorithm to task models that allow preemption, inter-task dependencies due to shared resources, and inter-task precedence relationships. Another aspect of future study include accounting for second-order main memory effects that limit decreases in task execution times with increases in processor frequency. Finally, using PA-BTA as a power-aware, real-time scheduling node in distributed real-time environments is also the subject of ongoing work.

7. Acknowledgements

This work was supported by the U.S. Office of Naval Research under Grant N00014-00-1-0549. We thank Mr. Dakai Zhu of University of Pittsburgh for his many insightful discussions that have helped improve the paper.

References

- [1] K. Chen and P. Muhlethaler. A scheduling algorithm for tasks described by time value function. *Journal of Real-Time Systems*, 10(3):293–312, May 1996.
- [2] P. Chou, J. Liu, D. Li, and N. Bagherzadeh. Impacct: Methodology and tools for power-aware embedded systems. *Design Automation For Embedded Systems, Special Issue on Design Methodologies and Tools for Real-Time Embedded Systems*, April 2002.
- [3] R. K. Clark. *Scheduling Dependent Real-Time Activities*. PhD thesis, Carnegie Mellon University, 1990. CMU-CS-90-155.
- [4] D. M. D. Zhu, N. AbouGhazaleh and R. Melhem. Power aware scheduling for and/or graphs in multi-processor real-time systems. In *Proceedings of The IEEE International Conference on Parallel Processing*, August 2002.
- [5] R. Graybill and R. Melhem. *Power Aware Computing*. Kluwer Academic/Plenum Publishers, 2002.
- [6] E. D. Jensen. Asynchronous decentralized real-time computer systems. In *Real-Time Computing*, Proceedings of the NATO Advanced Study Institute. Springer Verlag, October 1992.
- [7] E. D. Jensen and B. Ravindran. Guest editor’s introduction to special section on asynchronous real-time distributed systems. *IEEE Trans. on Computers*, 51(8):881–882, August 2002.
- [8] W. Kim, D. Shin, H. Yun, J. Kim, and S. L. Min. Performance comparison of dynamic voltage scaling algorithms for hard real-time systems. In *Proceedings of The IEEE Real-Time and Embedded Technology and Applications Symposium*, September 2002.
- [9] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.
- [10] J. Liu, P. Chou, and N. Bagherzadeh. Communication speed selections for embedded systems with networked voltage-scalable processors. In *Proceedings of The International Symposium on Hardware/Software Codesign*, April 2002.
- [11] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, New Jersey, 2000.
- [12] T. Martin. *Balancing Batteries, Power and Performance: System Issues in CPU Speed-Setting for Mobile Computing*. PhD thesis, Carnegie Mellon University, August 1999.
- [13] T. Martin and D. Siewiorek. Non-ideal battery and main memory effects on cpu speed-setting for low power. *IEEE Transactions on VLSI Systems*, 9(1):29–34, February 2001.
- [14] J. Wang. Soft real-time switched ethernet: Best-effort packet scheduling algorithm, implementation, and feasibility analysis. Master’s thesis, Virginia Tech, September 2002.
- [15] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proceedings of The USENIX Symposium on Operating Systems Design and Implementation*, pages 13–23, November 1994.