Energy-Efficient Scheduling of Periodic Real-Time Tasks over Homogeneous Multiprocessors

Jian-Jia Chen and Tei-Wei Kuo
Department of Computer Science and Information Engineering,
Graduate Institute of Networking and Multimedia,
National Taiwan University, Taipei, Taiwan, ROC.
Email:{r90079, ktw}@csie.ntu.edu.tw

Abstract

Different from many previous energy-efficient scheduling studies, this paper explores energy-efficient multiprocessor scheduling of periodic real-time tasks with different power consumption functions. When the goal is on the minimization of energy consumption, we propose a 1.412*-approximation algorithm in the derivation of a feasible schedule. A series of simulation experiments was done for the performance evaluation of the proposed algorithm.*

1 Introduction

With the advance technology of VLSI circuit designs, many modern processors, such as the Intel StrongARM SA1100 processor [19] and the Intel XScale [20], could now operate at various supply voltages and have different processor speeds. The power consumption of processors is usually a convex and increasing function of processor speeds, which is highly dependent on the hardware designs. The lower the speed, the less the power consumption is, where a lower processor speed usually means longer execution time for tasks.

In the past decades, energy-efficient task scheduling with various deadline constraints has received a lot of attention. Although many studies have been done for uniprocessor scheduling, such as [4, 6, 10, 11, 17, 25], not much work has been done for multiprocessor scheduling. As pointed out in [2], implementations of real-time systems with multiple processors could be often much more energy-efficient than those with a single processor, because of the convexity of power consumption functions. Due to the $\mathcal{NP}$-hardness of many multiprocessor energy-efficient scheduling problems, various heuristics were proposed in the derivation of schedules for different task models with an objective in the minimization of energy consumption, e.g., [1, 5, 8, 9, 12, 13, 18, 24, 26, 27]. In particular, several energy-efficient scheduling algorithms based on list heuristics were proposed [12, 13, 26]. Heuristic algorithms for periodic tasks in multiprocessor environments were proposed in [1, 5]. Zhu, et al. [27] explored on-line task scheduling with reclamation of slacks resulted from early completion of tasks during the run time. Mishra, et al. [18] explored energy-efficient scheduling issues with the considerations of the communication delay of tasks. In addition to the considerations of energy-efficient scheduling, Anderson and Sanjoy [2] explored the tradeoff between the total energy consumption of task executions and the number of required processors, where tasks in the proposed solutions run at the same speed. So far, not much work is done with approximation ratios in energy-efficient multiprocessor real-time scheduling. An example result is the approximation algorithms proposed for the scheduling of frame-based tasks in [8], where tasks share the same power consumption function, and [9], where tasks might have different power consumption functions. Energy-efficient multiprocessor scheduling of frame-based task sets was also explored in [24] for chip-multiprocessor (CMP) architectures, in which cores, i.e., processors, on a chip must share the same processor speed at any given time moment.

This paper considers energy-efficient scheduling of periodic real-time tasks over multiple processors. Different from previous energy-efficient scheduling studies, this research explores energy-efficient multiprocessor scheduling for periodic real-time tasks, in which each task might have different periods, initial arrival times, CPU execution cycles, and power consumption functions. The power consumption functions of tasks are modeled as $h \cdot s^{\alpha}$ [6, 14, 25], where α is a hardware-dependent factor, and h is a parameter related to the task under executions (Please see the discussions of power consumption functions in the next section). When the goal is on the minimization of energy consumption, we propose an approximation algorithm with an approximation ratio $\frac{(\alpha-1)^{\alpha-1}(2^{\alpha}-1)^{\alpha}}{\alpha^{\alpha}(2^{\alpha}-2)^{\alpha-1}}$, which is bounded by 1.412 since the value of α is at most 3 [6, 14, 25], in the

derivation of a feasible schedule. Simulation results show that our proposed algorithm not only guarantees the approximation factors but also derives solutions close to optimal solutions.

The rest of this paper is organized as follows: In Section 2, we define the system models and the multiprocessor energy-efficient scheduling problem. Section 3 presents an approximation algorithm for the multiprocessor energy-efficient scheduling problem. Section 4 presents evaluation results. Section 5 is the conclusion.

2 Models and Problem Definitions

2.1 Processor Models

We are interested in energy-efficient scheduling over homogeneous multiprocessors, where the power consumption function of each task remains the same for every processor. The power consumption function $P()$ in the dynamic voltage circuits is defined as a function of the adopted processor speed s [7, 23]:

$$P(s) = C_{ef} V_{dd}^2 s, \quad (1)$$

where $s = k\frac{(V_{dd}-V_t)^2}{V_{dd}}$, and C_{ef}, V_t, V_{dd}, and k denote the effective switch capacitance, the threshold voltage, the supply voltage, and a hardware-design-specific constant, respectively ($V_{dd} \geq V_t \geq 0$, $k > 0$, and $C_{ef} > 0$). The value of the effective switch capacitance is highly related to the software implementation and the execution path of a task (usually derived by profiling). Note that the power consumption function is a convex and increasing function of processor speeds. When V_t is 0, the power consumption function $P(s)$ could be rephrased as a cubic function of the processor speed s. As reported in the literature, e.g., [6, 14, 25], the power consumption function can be phrased as $h \cdot s^\alpha$, where α is a hardware-dependent factor, and h is a parameter related to the task under executions.

In this study, we assume that each processor could operate at any speed in $[0, \infty]$, and the speed of each processor could be adjusted independently from each another. We assume that the number of CPU cycles executed in a time interval is linearly proportional to the processor speed, and that the energy consumed for a processor in the execution of a task at the processor speed s for t time units is the multiplication of t and its corresponding power consumption $P(s)$ at the speed s. Let the amount of CPU cycles completed for a task running at a speed s for t time units be the multiplication of s and t. Suppose that the time and energy overheads required on speed/voltage switching be negligible.

2.2 Task Models

Tasks under discussions in this paper are periodic and independent in executions. A periodic task is an infinite sequence of task instances, referred to as *jobs*, where each job of a task comes in a regular period [15, 16]. Each task τ_i is associated with its initial arrival time (denoted by a_i), its execution CPU cycles (denoted by c_i), its period (denoted by p_i), and its power consumption function (denoted by $P_i()$). Note that c_i denotes the maximum number of CPU cycles required to complete the execution of any job of τ_i. The power consumption function $P_i()$ of each task τ_i is rephrased as a convex and increasing function of the processor speed s, i.e., $P_i(s) = h_i \cdot s^\alpha$, where α is a hardware-dependent constant between 2 and 3 [17, 21], and h_i is a positive parameter characterizing the average switch capacitance and the hardware factor. It is clear that each $P_i(s)$ is second-order differentiable. Given a set $\mathbf{T}$ of tasks, the *hyper-period* of $\mathbf{T}$, denoted by L, is defined as the least common multiple (LCM) of the periods of tasks in $\mathbf{T}$. Let the relative deadline of each task τ_i be equal to its period p_i in this paper. That is, the arrival time and deadline of the j-th job of task τ_i are $a_i + (j-1) \cdot p_i$ and $a_i + j \cdot p_i$, respectively.

2.3 Problem Definitions

A *schedule* of a task set $\mathbf{T}$ is a mapping of the executions of tasks in $\mathbf{T}$ to processors in the system with an assignment of a processor speed for each corresponding task execution, where the job arrivals of each task $\tau_i \in \mathbf{T}$ satisfy its timing constraints a_i and p_i. A schedule is *feasible* if no job misses its deadline, and all jobs of the same task execute on the same processor. The energy consumption of a schedule S, denoted as $\Phi(S)$, is the sum of the energy consumption of the executions of jobs in S. We are interested in real-time energy-efficient scheduling of independent tasks over multiple processors, where no task migration is allowed:

Definition 1 *The Minimization Problem of the Energy Consumption for Multiprocessor Scheduling*

Given a set $\mathbf{T}$ of independent tasks executing over M identical processors, the objective is to find a feasible schedule S for $\mathbf{T}$ in its hyper-period such that $\Phi(S)$ is minimized. □

Suppose that jobs of each task τ_i in a given schedule S execute at a speed s_i. $\Phi(S)$ is equal to $\sum_{\tau_i \in \mathbf{T}} \frac{L}{p_i} P_i(s_i) \frac{c_i}{s_i}$, where $\mathbf{T}$ is a given set of tasks under considerations, and L is the hyper-period of $\mathbf{T}$.

Theorem 1 *The Minimization Problem of the Energy Consumption for Multiprocessor Scheduling is $\mathcal{NP}$-hard.*

Proof. The correctness of this theorem follows from the fact that the corresponding problems, when $P_i(s) = s^3$, $a_i = 0$, and $p_i = D$, are $\mathcal{NP}$-hard (A similar argument to the proof in [8, Theorem 1]). □

With the $\mathcal{NP}$-hardness of the above problems, the objective of this research is to propose approximated solutions

with approximation bounds. Formally, a γ-approximation algorithm for the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling is an algorithm that derives a feasible schedule with an amount of energy consumption no more than γ times of an optimal solution (based on the definition of γ approximation in [22, §1]).

3 On the Minimization Problem of the Energy Consumption

In this section, we propose an approximation algorithm for the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling. If the number of tasks in $\mathbf{T}$ is no more than M, an optimal schedule would execute each task τ_i on a different processor at the speed c_i/p_i, for $i = 1, \ldots, |\mathbf{T}|$. For the rest of this section, we will focus our discussions on cases, where the number of tasks in $\mathbf{T}$ is more than M.

Let S be a feasible schedule of $\mathbf{T}$ for the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling. Let S_m denote the partial schedule of S on the m-th processor by removing the tasks running on the other processors, and T_m denote the set of tasks assigned to execute on the m-th processor. Note that $\cup_{m=1}^{M} T_m = \mathbf{T}$ and $T_m \cap T_n = \emptyset$ for any $m \neq n$. We claim that there must exist an optimal schedule S^* that satisfies the following two properties for any partial schedule S_m^* of S^*, where $1 \leq m \leq M$: (1) For every task τ_i in T_m^*, all jobs of τ_i execute at a common processor speed. (2) The total utilization tasks in S_m^*, which is defined as the sum of the utilization of each task (i.e., its execution time divided by its period) in S_m^*, is equal to 100%. This claim could be proved based on the convexity of power consumption functions by a similar argument to that for optimal energy-efficient scheduling in a uniprocessor system [4].

Let x_{im} be a binary variable to indicate whether τ_i is assigned to execute on the m-th processor, and t_i be a variable denoting the execution time of task τ_i. We can re-formulate the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling as a convex programming as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{\tau_i \in \mathbf{T}} E_i(t_i) \\ \text{subject to} & \sum_{\tau_i \in \mathbf{T}} x_{im} \cdot t_i/p_i = 1, \text{ for } \quad m = 1, \ldots, M \\ & t_i > 0, \quad \forall \tau_i \in \mathbf{T}, \\ & \sum_{m=1}^{M} x_{im} = 1, \quad \forall \tau_i \in \mathbf{T}, \text{ and} \\ & x_{im} \in \{0, 1\}, \quad \forall m = 1, \ldots, M, \text{ and } \tau_i \in \mathbf{T}, \end{array}$$

where $E_i(t_i)$ is defined as the *energy consumption* to execute all of the jobs of τ_i in the hyper-period L at the speed $\frac{c_i}{t_i}$, i.e., $E_i(t_i) = \frac{L}{p_i} P_i(\frac{c_i}{t_i}) t_i = \frac{L h_i}{p_i} \frac{c_i^{\alpha}}{t_i^{\alpha-1}}$. The reason why $t_i > 0$ comes from the assumption that the available speeds are continuous in $[0, \infty]$.

Our proposed algorithm for the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling consists of two phases: the relaxation phase and the rounding phase. In the relaxation phase, we relax the integral constraints on the variables x_{im} and derive an optimal solution for the relaxed problem (which is a lower bound on the energy consumption of an optimal schedule). In the rounding phase, we derive a feasible schedule based on the solution derived in the first phase.

3.1 Relaxation Phase

With the integral constraints on x_{im} being relaxed, we could first rewrite the above convex programming problem as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{\tau_i \in \mathbf{T}} E_i(t_i), \\ \text{subject to} & \sum_{\tau_i \in \mathbf{T}} t_i/p_i = M, \text{ and} \\ & 0 < t_i \leq p_i. \end{array} \tag{2}$$

An optimal solution for Equation (2) is a lower bound on the energy consumption for optimal schedules for $\mathbf{T}$ in the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling. Equation (2) can be resolved by applying the Karush-Kuhn-Tucker optimality condition in $O(|\mathbf{T}| \log |\mathbf{T}|)$. (Detail procedures to derive an optimal solution of Equation (2) can be found in [3, 4, 9].) Let $(t_1^*, t_2^*, \ldots, t_{|\mathbf{T}|}^*)$ be an optimal solution for Equation (2).

Lemma 1 *When $t_i^* < p_i$ and $t_j^* < p_j$, $p_i E_i'(t_i^*) = p_j E_j'(t_j^*)$, where $E_i'()$ and $E_j'()$ are the derivatives of $E_i()$ and $E_j()$, respectively.*

Proof. This Lemma is based on the Karush-Kuhn-Tucker condition for the optimal solution $(t_1^*, t_2^*, \ldots, t_{|\mathbf{T}|}^*)$, in which $E_i'(t_i^*) - \frac{\lambda}{p_i} = 0$, and $E_j'(t_j^*) - \frac{\lambda}{p_j} = 0$ for some constant λ when $t_i^* < p_i$ and $t_j^* < p_j$. □

3.2 Rounding Phase

Let the utilization $u_i^* = t_i^*/p_i$ of task τ_i in $\mathbf{T}$ derived in the first phase be called the *estimated utilization* of τ_i. In this phase, we derive a feasible schedule based on the estimated utilizations of the tasks derived in the first phase, i.e., $(u_1^*, u_2^*, \ldots, u_{|\mathbf{T}|}^*)$, by adopting the *Largest-Estimated-Utilization-First* strategy. The proposed algorithm is shown in Algorithm 1 and denoted as Algorithm LEUF:

Let T_m denote the set of the tasks assigned to execute on the m-th processor, which is an empty set initially. U_m denotes the *total estimated utilization* on the m-th processor, which is defined as the sum of the estimated utilizations of tasks in T_m. Tasks are considered to execute on a selected processor in a non-increasing order of their estimated utilizations. A task under consideration is assigned to execute on the m-th processor with the smallest total estimated utilization U_m (Tie-breaking is done by choosing the smallest index m). After all of the tasks in $\mathbf{T}$ are assigned to execute

on a specific processor, the utilization of τ_i is set as $\frac{u_i^*}{U_m}$ for every task τ_i in T_m. That is, the execution time of every job of task τ_i is set as $\frac{t_i^*}{U_m}$. The transformation of job execution times would result in a situation in which the total utilization of tasks assigned on a processor is exactly equal to 100%. The scheduling of tasks on each processor could be done successfully by the earliest-deadline-first scheduling algorithm because the earliest-deadline-first scheduling algorithm could always schedule periodic real-time independent tasks with a total utilization no more than one [15]. The time complexity of Algorithm LEUF is $O(|\mathbf{T}| \log |\mathbf{T}|)$. For the simplicity of representation, any schedule derived by Algorithm LEUF is denoted as S_{LEUF}.

Algorithm 1 : LEUF

Input: $(\mathbf{T}, M)$;
Output: A feasible schedule;
1: **if** $|\mathbf{T}| \leq M$ **then**
2: return the schedule by executing each task τ_i in $\mathbf{T}$ at the speed $\frac{c_i}{p_i}$ on the i-th processor;
3: let u_i^* be the estimated utilization for $\tau_i \in \mathbf{T}$;
4: sort $\mathbf{T}$ in a non-increasing order of their estimated utilizations;
5: $U_1 \leftarrow U_2 \leftarrow \cdots \leftarrow U_M \leftarrow 0$, and $T_1 \leftarrow T_2 \leftarrow \cdots \leftarrow T_M \leftarrow \emptyset$;
6: **for** $i = 1$ to $|\mathbf{T}|$ **do**
7: find the smallest U_m; (break ties by choosing the smallest index m)
8: $T_m \leftarrow T_m \cup \{\tau_i\}$ and $U_m \leftarrow U_m + u_i^*$;
9: **for** $m = 1$ to M **do**
10: **for each** task $\tau_i \in T_m$ **do**
11: $t_i' \leftarrow t_i^* \times \frac{1}{U_m}$;
12: return the schedule S_{LEUF} which executes task τ_i in T_m ($1 \leq m \leq M$) at the speed c_i/t_i' on the m-th processor in an earliest-deadline-first order;

3.3 Analysis of the Approximation Ratio

For notational brevity, let e_i^* be the *estimated energy consumption* of the jobs of task τ_i in the hyper-period, i.e., $e_i^* = E_i(t_i^*)$. Let $\mathbf{T}'$ be the subset of $\mathbf{T}$, where $\mathbf{T}'$ consists of tasks whose estimated utilizations are all strictly less than 1. That is, $\mathbf{T}' = \{\tau_i \mid t_i^*/p_i < 1, \forall \tau_i \in \mathbf{T}\}$. For notational brevity, let $\hat{\mathbf{T}}$ be $\mathbf{T} \setminus \mathbf{T}'$. Note that we only focus our discussions on the case that $\mathbf{T}'$ is not empty, since Algorithm LEUT guarantees to derive an optimal schedule for the other case.

Lemma 2 *For any two tasks $\tau_i, \tau_j \in \mathbf{T}'$, $\frac{e_i^*}{u_i^*} = \frac{e_j^*}{u_j^*}$.*

Proof. By the equality of $h_i \frac{L}{p_i} \frac{c_i^\alpha}{(t_i^*)^\alpha} \cdot p_i = \frac{L}{p_j} h_j \frac{c_j^\alpha}{(t_j^*)^\alpha} \cdot p_j$ in Lemma 1, we know that $\frac{u_i^*}{u_j^*} = \frac{e_i^*}{e_j^*}$. □

Lemma 3 *Suppose that U_{m^*} and $U_{\hat{m}}$ are the maximum and the minimum total utilizations, respectively, then $U_{\hat{m}} \leq 1 \leq U_{m^*} \leq 2U_{\hat{m}}$.*

Proof. By definition, we know that $U_{\hat{m}} \leq 1 \leq U_{m^*}$. If U_{m^*} is equal to 1, we know that $U_{\hat{m}}$ is also equal to 1 by applying the pigeon-hole principle. For the rest of this discussion, we only focus on the other case that U_{m^*} is greater than 1. Since the estimated utilization of a task is no greater than 1, T_{m^*} consists of at least two tasks. Let τ_v be the last one inserted into T_{m^*}. Since the tasks are assigned in a non-increasing order of their estimated utilization to execute on the processor whose current total estimated utilization is the smallest, we know $u_v^* \leq U_{m^*} - u_v^* \leq U_{\hat{m}}$. Therefore, we have $U_{m^*} \leq 2U_{\hat{m}}$. □

Lemma 4 *Suppose $f(x) = k \cdot (2x)^\alpha + (H - k)x^\alpha$ for a positive number H and a non-negative number k, where $0 \leq k \leq H$ and $2k \cdot x + (H - k) \cdot x = H$, then*

$$f(x) \leq \frac{(\alpha - 1)^{\alpha-1}(2^\alpha - 1)^\alpha}{\alpha^\alpha (2^\alpha - 2)^{\alpha-1}} H.$$

Proof. Since $2k \cdot x + (H - k) \cdot x = H$, we know $k = \frac{H - Hx}{x}$. Therefore,

$$f(x) \quad = H(x^{\alpha-1}(2^\alpha - 1) + x^\alpha(2 - 2^\alpha)),$$

and the derivative of $f(x)$ is

$$f'(x) = H((\alpha - 1)x^{\alpha-2}(2^\alpha - 1) + \alpha x^{\alpha-1}(2 - 2^\alpha)).$$

$f(x)$ is maximized at x^* when $f'(x^*) = 0$. By solving $f'(x^*) = 0$, we have $x^* = \frac{(\alpha-1)(2^\alpha-1)}{\alpha(2^\alpha-2)}$. As a result, we conclude that $f(x) \leq f(x^*) = \frac{(\alpha-1)^{\alpha-1}(2^\alpha-1)^\alpha}{\alpha^\alpha(2^\alpha-2)^{\alpha-1}} H$. □

Based on Lemmas 2, 4, and 3, the approximation ratio of the algorithm could be proved as follows:

Theorem 2 *Algorithm* LEUT *is a polynomial-time $\frac{(\alpha-1)^{\alpha-1}(2^\alpha-1)^\alpha}{\alpha^\alpha(2^\alpha-2)^{\alpha-1}}$-approximation algorithm for the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling.*

Proof. Let τ_r be a task in $\mathbf{T}'$. Based on Lemma 2 and the optimality of $\sum_{\tau_i \in \mathbf{T}} e_i^*$, we have $\Phi(S^*) \geq \sum_{\tau_i \in \mathbf{T}} e_i^* = \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + e_r^*/u_r^* \sum_{\tau_i \in \mathbf{T}'} u_i^* = \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + e_r^*/u_r^*(M - |\hat{\mathbf{T}}|)$, where S^* is an optimal schedule for $\mathbf{T}$.

Since u_i^* is equal to 1 for $1 \leq i \leq |\hat{\mathbf{T}}|$, the i-th processor is assigned only a task in S_{LEUF} . Based on Lemma 2, we have

$$\Phi(S_{\text{LEUF}}) = \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{m=|\hat{\mathbf{T}}|+1}^{M} \frac{e_r^*}{u_r^*} (U_m)^\alpha. \qquad (3)$$

The approximation ratio $\mathcal{A}$ of Algorithm LEUF is

$$\mathcal{A} = \frac{\Phi(S_{\text{LEUF}})}{\Phi(S^*)} \leq \frac{\sum_{m=|\hat{\mathbf{T}}|+1}^{M} (U_m)^\alpha}{M - |\hat{\mathbf{T}}|}. \qquad (4)$$

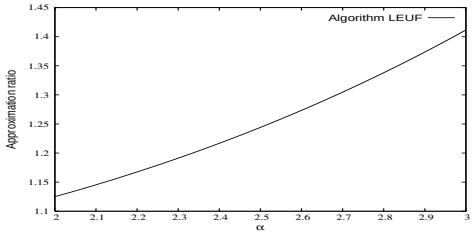


Figure 1. The approximation ratio of Algorithm LEUF for different values of α.

Based on Lemma 3, we have $2U_{\hat{m}} \geq U_{m^*} \geq U_m \geq U_{\hat{m}}$, for all $|\hat{\mathbf{T}}| < m \leq M$. Because of the convexity of the function U_m^α of U_m and the fact $2U_{\hat{m}} - U_m \geq 0$, we have

$$\sum_{m=|\hat{\mathbf{T}}|+1}^{M} U_m^\alpha \leq k \cdot (2U_{\hat{m}})^\alpha + (M - |\hat{\mathbf{T}}| - k)(U_{\hat{m}})^\alpha,$$

where $2k \cdot U_{\hat{m}} + (M - |\hat{\mathbf{T}}| - k)U_{\hat{m}} = (M - |\hat{\mathbf{T}}|)$. Let $f(x)$ be defined as $k \cdot (2x)^\alpha + (H - k)x^\alpha$ for a positive number H and a non-negative number k, where $k \leq H$ and $2k \cdot x + (H - k) \cdot x = H$. By Lemma 4, $f(x) \leq \frac{(\alpha-1)^{\alpha-1}(2^\alpha-1)^\alpha}{\alpha^\alpha(2^\alpha-2)^{\alpha-1}} H$ by solving $f'(x) = 0$. By setting H as $(M - |\hat{\mathbf{T}}|)$ and considering Equation (4), this theorem is proved. □

Corollary 1 *The approximation ratio of Algorithm* LEUF *is* 1.412.

Proof. The proof is done by setting α as 3. □

For different values of α, the approximation ratio of Algorithm LEUF is illustrated in Figure 1.

4 Performance Evaluation

In this section, we provide performance evaluation on the energy consumption of Algorithm LEUF. We also implemented algorithms in [1, 5], and revised the algorithm [8] by sorting tasks in a non-increasing order of c_i/p_i. However, the performace of these algorithms was much worse than Algorithm LEUF since they were proposed for tasks with the same power consumption function. Hence, another algorithm, denoted as Algorithm RAND, which is very similar to Algorithm LEUF, was simulated for comparison. The only difference between Algorithm RAND and Algorithm LEUF is that tasks are not sorted before the assignment procedure in Algorithm RAND.

4.1 Workload Parameters and Performance Metrics

Each periodic real-time task was generated based on three parameters: the number b_i of jobs within the time interval L, the required CPU cycles c_i, and the coefficient h_i of the power consumption function. The value of b_i was an integral variable uniformly distributed in the range of $[1, 16]$. c_i was an integral variable uniformly distributed in the range of $[1, 100]$, while h_i was uniformly distributed in the range of $[2, 10]$. The exponent of the power consumption functions of the processor speed s was set as 3, i.e., $P_i(s) = h_i s^3$, provided that the threshold voltage V_t is 0. To evaluate the effect of the exponent of the power consumption function, we also perform simulations by setting α as a random variable between 2.5 and 3 used for a set of tasks under simulations. The period of task τ_i was set as $\frac{L}{b_i}$.

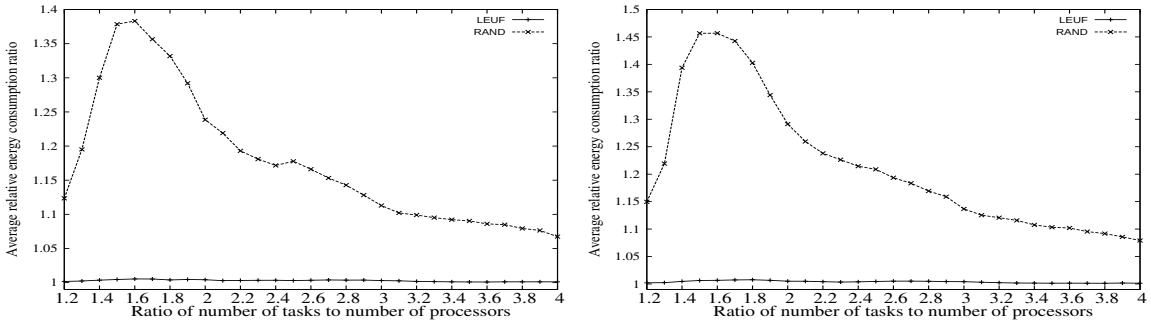
We simulated the algorithms for the effects on the ratio of the number of tasks to the number of processors. For a given ratio η of the number of tasks to the number of processors, the number of processors M was an integral random variable between 10 and 30, and the number of tasks was set as the floor of the multiplication of η and M, i.e., $\lfloor \eta \cdot M \rfloor$. The *relative energy consumption ratio* was adopted as the performance metric in our experiments. The relative energy consumption ratio for an input instance was defined as the energy consumption of the schedule derived by the algorithm to the optimal solution of Equation (2).

4.2 Experimental Results

For the Minimization Problem of the Energy Consumption for Multiprocessor Scheduling, Figures 2(a) and 2(b) present the average relative energy consumption ratios for the simulated algorithms when α is in the range of $[2.5, 3]$ and is 3, respectively. The performance of Algorithm LEUF was very close to that of the optimal solutions. The average relative energy consumption ratios for Algorithm LEUF were less than 1.01. The average relative energy consumption ratios for Algorithm RAND were less than 1.46. When the ratio of the number of tasks to the number of processors was small, both of Algorithm LEUF and Algorithm RAND might assign a task along with improper tasks on a processor. Such an assignment might result in a significant increase on the energy consumption of these tasks when the energy consumption for the other tasks were almost as the same as that in the optimal schedule. When the ratio of the number of tasks to the number of processors was small, in most cases, most processors were assigned with only one task, and the assignment was almost as the same as that of an optimal schedule. Therefore, the average energy consumption ratio was relatively small when the ratio of the number of tasks to the number of processors was less than 1.6.

5 Conclusion

In this paper, we explore approximation algorithms for energy-efficient scheduling of periodic real-time tasks over



(a) Average ratio when α is in the range of 2.5 and 3

(b) Average ratio when $\alpha = 3$

Figure 2. (a) and (b): average relative energy consumption ratios for different settings on α.

multiple processors, where the scheduling problem is $\mathcal{NP}$-hard. The task model explored in this work is more general than many previous studies in energy-efficient multiprocessor real-time scheduling, where tasks under considerations might have different periods, initial arrival times, CPU execution cycles, and power consumption functions. When the goal is on the minimization of energy consumption, we propose a 1.412-approximation algorithm in the derivation of a feasible schedule.

References

[1] T. A. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *Proceedings of the 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'05)*, pages 213–223, 2005.

[2] J. H. Anderson and S. K. Baruah. Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, pages 428–435, 2004.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez. Optimal reward-based scheduling for periodic real-time tasks. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99)*, pages 79–89, 1999.

[4] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proceedings of the IEEE EuroMicro Conference on Real-Time Systems*, pages 225–232, 2001.

[5] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Proceedings of 17th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 113 – 121, 2003.

[6] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *Proceedings of the 2004 Symposium on Foundations of Computer Science*, pages 520–529, 2004.

[7] A. Chandrakasan, S. Sheng, and R. Broderson. Lower-power CMOS digital design. *IEEE Journal of Solid-State Circuit*, 27(4):473–484, 1992.

[8] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, and T.-W. Kuo. Multiprocessor energy-efficient scheduling with task migration considerations. In *EuroMicro Conference on Real-Time Systems (ECRTS'04)*, pages 101–108, 2004.

[9] J.-J. Chen and T.-W. Kuo. Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics. In *International Conference on Parallel Processing (ICPP)*, pages 13–20, 2005.

[10] J.-J. Chen, T.-W. Kuo, and H.-I. Lu. Power-saving scheduling for weakly dynamic voltage scaling devices. In *Workshop on Algorithms and Data Structures (WADS)*, pages 338–349, 2005.

[11] J.-J. Chen, T.-W. Kuo, and C.-L. Yang. Profit-driven uniprocessor scheduling with energy and timing constraints. In *ACM Symposium on Applied Computing*, pages 834–840, 2004.

[12] F. Gruian. System-level design methods for low-energy architectures containing variable voltage processors. In *Power-Aware Computing Systems*, pages 1–12, 2000.

[13] F. Gruian and K. Kuchcinski. Lenes: Task scheduling for low energy systems using variable supply voltage processors. In *Proceedings of Asia South Pacific Design Automation Conference*, pages 449–455, 2001.

[14] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 37–46. Society for Industrial and Applied Mathematics, 2003.

[15] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[16] J. W. Liu. *Real-Time Systems*. Prentice Hall, Englewood, Cliffs, NJ., 2000.

[17] P. Mejía-Alvarez, E. Levner, and D. Mossé. Adaptive scheduling server for power-aware real-time tasks. *ACM Transactions on Embedded Computing Systems*, 3(2):284–306, 2004.

[18] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *International Parallel and Distributed Processing Symposium*, page 21, 2003.

[19] INTEL. Strong ARM SA-1100 Microprocessor Developer's Manual, 2003. INTEL.

[20] INTEL-XSCALE, 2003. http://developer.intel.com/design/xscale/.

[21] Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference*, pages 134–139. ACM Press, 1999.

[22] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[23] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proceedings of Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.

[24] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In *Proceedings of the 8th Conference of Design, Automation, and Test in Europe (DATE)*, pages 468–473, 2005.

[25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382. IEEE, 1995.

[26] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Annual ACM IEEE Design Automation Conference*, pages 183–188, 2002.

[27] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. In *Proceedings of IEEE 22th Real-Time System Symposium*, pages 84–94, 2001.