Approximation Algorithms for Multiprocessor Energy-Efficient Scheduling of Periodic Real-Time Tasks with Uncertain Task Execution Time*

Jian-Jia Chen
Computer Engineering
and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH)
Zurich, Switzerland
Email: chen@tik.ee.ethz.ch

Chuan-Yue Yang, Hsueh-I Lu, Tei-Wei Kuo
Department of Computer Science
and Information Engineering
National Taiwan University
Taipei, Taiwan
Email: {r92032, hil, ktw}@csie.ntu.edu.tw

Abstract

Energy-efficiency has been an important system issue in hardware and software designs for both real-time embedded systems and server systems. This research explores systems with probabilistic distribution on the execution time of real-time tasks on homogeneous multiprocessor platforms with the capability of dynamic voltage scaling (DVS). The objective is to derive a task partition which minimizes the expected energy consumption for completing all the given tasks in time. We give an efficient 1.13*-approximation algorithm and a polynomial-time approximation scheme (PTAS) to provide worst-case guarantees for the strongly* $\mathcal{NP}$*-hard problem. Experimental results show that the algorithms can effectively minimize the expected energy consumption.*

Keywords: Dynamic Voltage Scaling (DVS), Multiprocessor Scheduling, Probability, Expected Energy Consumption Minimization, Energy-Efficient Scheduling.

1 Introduction

With the advancements in VLSI circuit designs, modern processors can operate dynamically at different supply voltages, which lead to different execution speeds/frequencies. Well-known examples for embedded systems are Intel StrongARM SA1100 and Intel XScale. Technologies, such as Intel SpeedStep® and AMD PowerNOW!™, provide dynamic voltage scaling (DVS) for computer systems to prolong battery life. In the past decade, energy-efficient task scheduling has received a lot of attention. Many studies, such as [4, 16, 30], explore DVS scheduling to minimize the energy consumption when the tasks/jobs are executed in their worst cases.

In addition to worst-case estimations, profiling can also help system designers get the distribution information of the workload of a task. Given the probability distribution of workload, some previous studies [5,11,13,18,19,26,27,31,33] provide DVS scheduling strategies to reduce the *expected* energy consumption. Lorch and Smith [18] derived an accelerating frequency scheduling by executing a task at a lower frequency at the beginning and at higher frequencies for the rest, while concurrent tasks were treated as joint workload. Gruian [11] considered the scheduling of multiple tasks and allocated execution time to tasks based on their worst-case execution cycles. Yuan and Nahrstedt [31] exploited the accelerating scheduling strategy for soft real-time multimedia tasks. Xu et al. [26, 27], Zhang et al. [33], and Lu et al. [19], and Chen [5] explored inter-task scheduling for frame-based real-time tasks, in which all tasks have a common deadline and arrive at the same time.

Moreover, implementations of real-time systems with multiple processors are often more energy-efficient than those with a single processor because of the convexity of power consumption functions [2]. Various heuristics have been proposed for energy consumption minimization under different task and processor models in multiprocessor environments [1, 3, 6–8, 12, 14, 15, 20, 24, 25, 29, 32].

This paper explores task partition and scheduling for the minimization of expected energy consumption in homogeneous multiprocessor systems with the capability of dynamic voltage scaling. The objective is to minimize the expected energy consumption for completing all the given tasks in time. This problem was first explored by Xian, Lu, and Li [25], in which a heuristic algorithm was proposed by applying a variation of the worst-fit decreasing bin packing algorithm for balancing load with respect to a mathematical parameter related to expected energy consumption. Distinct from heuristic approaches, this paper gives polynomial-time approximation algorithms for the strongly $\mathcal{NP}$-hard problem to provide worst-case guarantees in the expected energy consumption of the derived solutions.

The dynamic (or speed-dependent) power consumption function, here, is modeled as s^α, where s is the processor speed and α is a hardware-dependent factor between 1 and 3. We show that an extension of the load-balancing approach suggested by Xian, Lu, and Li [25] with $O(|\mathbf{T}| \log |\mathbf{T}|)$ time complexity is a $\frac{(\alpha-1)^{\alpha-1}(3^\alpha-2^\alpha)^\alpha}{(2\cdot 3^\alpha-3\cdot 2^\alpha)^{\alpha-1}\alpha^\alpha}$-approximation algorithm, where $\mathbf{T}$ is the set of the given real-time tasks. Since α is at most 3, the approximation ratio is at most 1.13. In addition to the derivation of the approximation ratio, we also give the physical meaning, i.e., estimated utilization, of the load-

*Support in parts by research grants from ROC National Science Council NSC-96-2752-E-002-008-PAE.

balancing approach to see why it works, while only mathematical meaning was provided by Xian, Lu, and Li [25]. Moreover, by rounding the estimated worst-case utilization of tasks, we develop a polynomial-time approximation scheme (PTAS) to provide a $(1 + \zeta)$-approximated solution for any $1 > \zeta > 0$ for such a strongly $\mathcal{NP}$-hard problem, which is the best achievement in the development of polynomial-time approximation algorithms unless $\mathcal{NP} = \mathcal{P}$. The proposed polynomial-time approximation scheme allows the system designer to trade the optimality of the derived solution with the analysis time. Experimental results show that the algorithms can effectively minimize the expected energy consumption and derive solutions with near-optimal performance.

The rest of this paper is organized as follows. Section 2 defines the system models and the problem under considerations. Section 3 presents the polynomial-time approximation algorithm and the approximation scheme for the studied problem. Section 4 shows the performance evaluation of the algorithms with respect to the expected energy consumption. Section 5 concludes this paper.

2 Systems models and problem definitions

Processor models We explore energy-efficient scheduling over M homogeneous DVS multiprocessors, where the power consumption function of each task is the same for every processor. The power consumption function $P(s)$ of the adopted processor speed s has two parts $P_d(s)$ and P_{ind}, where $P_d(s)$ (P_{ind}, respectively) is dependent (independent, respectively) on speed s. Leakage power consumption mainly contributes to P_{ind}, while the dynamic power consumption resulting from the charging/discharging of gates on a CMOS DVS processor and the short-circuit power consumption contribute to $P_d(s)$. The speed-dependent power consumption function $P_d(s)$ could be modeled as a convex and increasing function of speed s. For example, the dynamic power consumption $P_{switch}(s)$, which dominates the power consumption in function $P_d(s)$ for processors in micro-meter manufacturing, in CMOS DVS processors due to gate switching at speed s is

$$P_{switch}(s) = C_{ef} V_{dd}^2 s, \tag{1}$$

where $s = \kappa \frac{(V_{dd}-V_t)^2}{V_{dd}}$, and C_{ef}, V_t, V_{dd}, and κ denote the effective switch capacitance, the threshold voltage, the supply voltage, and a hardware-design-specific constant, respectively ($V_{dd} \geq V_t \geq 0$, $\kappa > 0$, and $C_{ef} > 0$) [21]. If the leakage power consumption is related to the speeds/voltages the leakage power consumption is divided into two parts that contribute to $P_d(s)$ and P_{ind} accordingly. In other words, $P_d(s)$ models the voltage-dependent power consumption while P_{ind} models the voltage-independent power consumption [15].

As shown in the literature, for example [4, 16, 30], the speed-dependent power consumption function can be phrased as s^α, where α is a hardware-dependent factor between 1 and 3. Therefore, $P_d(s)$ is a convex and increasing function of s. The number of cycles executed in an interval $(t_1, t_2]$ is $\int_{t_1}^{t_2} s(t)dt$, and the energy consumption is $\int_{t_1}^{t_2} P(s(t))dt$, where $s(t)$ is the speed at time t. Since the variation of execution speeds does not affect the energy consumption resulting from the speed-independent power consumption, we do not account for the speed-independent power consumption in our theoretical analysis for the clarity of presentation. Hence, the power consumption function $P(s)$ at speed s on a processor, here-after, is $P_d(s)$. All the algorithms and analysis in this paper can be adopted correctly when $P(s)$ is $P_d(s)$ + P_{ind}.

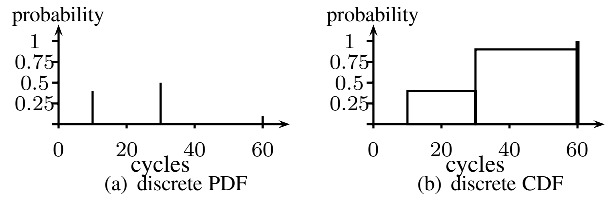


Figure 1. An example for the probability function of workload information of a task.

We assume that the speed of each processor can be adjusted independently, and each processor can ideally operate at any speed in $[0, \infty)$ implicitly. To cope with systems with an upper bound on the speeds, constraint violation approaches [17] might be used since deriving a feasible solution is $\mathcal{NP}$-complete. We will sketch the idea in Section 3.4.

Task models Tasks considered in this paper are periodic and independent in execution. 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. Each task τ_i is associated with its period (denoted as p_i) and its computation requirement in CPU cycles in the worst cases (denoted as c_i). The relative deadline of each task τ_i is equal to its period p_i. Given a task set $\mathbf{T}$, the *hyper-period* of $\mathbf{T}$, denoted by L, is defined as the minimum positive L so that L/p_i is an integer for any task τ_i in $\mathbf{T}$. For example, L is the least common multiple (LCM) of the periods of tasks in $\mathbf{T}$ when the periods of tasks are all integers. Note that the hyper-period is used only for the length of the time interval to evaluate the expected energy consumption. If it does not exist, we can use any value that is large enough with the same analytical results. Throughout the paper, we use the earliest-deadline-first (EDF) scheduling for task executions.

The computation requirement of task τ_i is profiled as a discrete probability density function (discrete PDF) for the number of execution cycles. For each task τ_i, the range $(0, c_i]$ is divided into β_i bins with different sizes. The j-th bin of task τ_i is associated with its amount of cycle $X_{i,j}$ and its probability density $\psi_i(j)$. Figure 1 illustrates an example of a task τ_i with $\beta_i = 3$, where $X_{i,1} = 10$, $X_{i,2} = 20$, and $X_{i,3} = 30$ with $\psi_i(1) = 0.4$, $\psi_i(2) = 0.5$, and $\psi_i(3) = 0.1$. Therefore, the probability for task τ_i with $\sum_{b=1}^{j} X_{i,b}$ cycles is $\psi_i(j)$. The discrete cumulative density function (CDF) for task τ_i to have cycles no more than $\sum_{b=1}^{j} X_{i,b}$ is $\Psi_i(j) = \sum_{b=1}^{j} \psi_i(b)$. By definition, $\Psi_i(\beta_i) = 1$ and $\sum_{b=1}^{\beta_i} X_{i,b} = c_i$. For notational brevity, we define $\Psi_i(0)$ as 0. Therefore, the probability that the schedule has to execute the first $\sum_{b=1}^{j} X_{i,b}$ cycles of task τ_i is $1 - \Psi_i(j-1)$, denoted by $\Psi_i^\dagger(j)$. Figure 1(a) shows the discrete PDF, while Figure 1(b) is the discrete CDF. Hence, $\Psi_i^\dagger(1) = 1$, $\Psi_i^\dagger(2) = 0.6$, and $\Psi_i^\dagger(3) = 0.1$ in the example.

Expected Energy Consumption The expected energy consumption, denoted by $\hat{E}_i(t_i)$, to complete a job of task τ_i in time amount t_i in the worst case can be derived by solving the following convex programming:

$$\begin{array}{ll} \text{minimize} & \sum_{b=1}^{\beta_i} s_{i,b}^{\alpha} \frac{X_{i,b}}{s_{i,b}} \Psi_i^{\dagger}(b) \\ \text{subject to} & \sum_{b=1}^{\beta_i} \frac{X_{i,b}}{s_{i,b}} \leq t_i \text{ and} \\ & s_{i,b} \geq 0,\ \forall b = 1, 2, \ldots, \beta_i, \end{array} \quad (2)$$

where $s_{i,b}$ is the speed to execute the computation requirement $X_{i,b}$. Equation (2) can be solved by applying the Lagrange Multiplier method, similarly to those in [25, 31]. The optimal solution of Equation (2) is to execute $X_{i,j}$ at speed $\frac{\sum_{b=1}^{\beta_i} X_{i,b} \sqrt[\alpha]{\Psi_i^{\dagger}(b)}}{t_i \sqrt[\alpha]{\Psi_i^{\dagger}(j)}}$ with expected energy consumption equal to $\frac{\left(\sum_{b=1}^{\beta_i} X_{i,b} \sqrt[\alpha]{\Psi_i^{\dagger}(b)}\right)^{\alpha}}{t_i^{\alpha-1}}$. For notational brevity, let h_i be $\left(\sum_{b=1}^{\beta_i} X_{i,b} \sqrt[\alpha]{\Psi_i^{\dagger}(b)}\right)^{\alpha}$. Hence, the optimal expected energy consumption $\hat{E}_i(t_i)$ to complete a job of task τ_i in time t_i is $\frac{h_i}{t_i^{\alpha-1}}$. Moreover, the expected energy consumption $E_i(t_i)$ in the hyper-period L for task τ_i is $\hat{E}_i(t_i)\frac{L}{p_i}$.

Note that if only one speed is allowed (to avoid too much speed switching) to execute task τ_i for a time interval with t_i time units, the expected energy consumption is $\frac{h_i}{t_i^{\alpha-1}}$, where h_i is $c_i^{\alpha-1} \sum_{b=1}^{\beta_i} X_{i,b} \Psi_i^{\dagger}(b)$ by executing task τ_i at speed $\frac{c_i}{t_i}$ [27].

Problem Definition A *schedule* of a task set $\mathbf{T}$ is a mapping of the executions (task partition) of tasks in $\mathbf{T}$ to processors in the system with an assignment of processor speeds for each corresponding task execution, where the job arrivals of each task $\tau_i \in \mathbf{T}$ satisfy its timing constraint p_i. A schedule is *feasible* if no job misses its deadline, and all jobs of the same task execute on the same processor. By applying EDF for scheduling, the problem can be formulated as the following programming:

$$\begin{array}{ll} \text{minimize} & \sum_{\tau_i \in \mathbf{T}} \hat{E}_i(t_i) \frac{L}{p_i} \\ \text{subject to} & \sum_{\tau_i \in \mathbf{T}} x_{im} \cdot t_i / p_i \leq 1, \text{ for } \quad m = 1, \ldots, M \\ & \sum_{m=1}^{M} x_{im} = 1, \quad \forall \tau_i \in \mathbf{T}, \text{ and} \\ & x_{im} \in \{0, 1\}, \quad \forall \tau_i \in \mathbf{T}, \text{ and } m = 1, \ldots, M, \end{array} \quad (3)$$

where x_{im} is a binary variable to indicate whether τ_i is assigned on processor m, t_i is a variable denoting the execution time of task τ_i, and $\hat{E}(t_i)$ is $\frac{h_i}{t_i^{\alpha-1}}$. We denote the problem as the *multiprocessor expected-energy-efficient scheduling* problem. The expected energy consumption of a schedule S is denoted by $\Phi(S)$. By an argument similar to [6, Theorem 1], the multiprocessor expected-energy-efficient scheduling problem is $\mathcal{NP}$-hard in a strong sense even for the special case with $P_i(s) = s^3$, $\beta_i = 1$, and $p_i = D$ for any fixed $D > 0$.

Due to the $\mathcal{NP}$-hardness of the problem, we focus the study on polynomial-time approximation algorithms with worst-case guarantees. For any input instance, a γ-approximation algorithm derives a solution with at most γ times of the expected energy consumption of an optimal solution, where γ is referred to as the *approximation ratio* of the algorithm. This paper provides a combinatorial approximation algorithm with low time complexity. Moreover, a polynomial-time approximation scheme (PTAS) is provided to have trade-offs between the user's tolerable approximation ratio and the complexity. An algorithm for a minimization problem is said to be a PTAS if (1) it is a $(1+\zeta)$-approximation algorithm, and (2) its time complexity is polynomial in the input size by treating ζ as a constant, where ζ is a positive user-input parameter in a specified range. For a $\mathcal{NP}$-hard problem in a strong sense, PTAS is the best achievement in approximation algorithms [23].

3 Polynomial-Time Approximation algorithms

This section presents polynomial-time approximation algorithms for the multiprocessor expected-energy-efficient scheduling problem. Our proposed algorithms consist of two phases. In the first phase, the relaxation phase, we relax the integral constraints on the variables x_{im} in Equation (3) and derive an optimal solution for the relaxed problem, which will be presented in Section 3.1. Then, a feasible schedule based on the optimal solution of the relaxed problem are derived in the second phase, the assigning phase. This paper presents two different algorithms with different approximation ratios in the assigning phase, in which one presented in Section 3.2 is more efficient and with a constant approximation ratio, and the other in Section 3.3 is with adjustable tradeoffs between the complexity and the approximation ratio.

If the number of tasks in $\mathbf{T}$ is no more than M, an optimal schedule would execute each task τ_i on an individual processor, for $i = 1, \ldots, |\mathbf{T}|$. For the rest of this section, we will focus on the other cases, where the number of tasks in $\mathbf{T}$ is more than M. Let S be a feasible schedule of $\mathbf{T}$ for the multiprocessor expected-energy-efficient scheduling problem. Let S_m denote the partial schedule of S on processor m, and $\mathbf{T}_m$ denote the set of tasks assigned to execute on processor m. Hence, $\cup_{m=1}^{M} \mathbf{T}_m = \mathbf{T}$ and $\mathbf{T}_m \cap \mathbf{T}_n = \emptyset$ for any $m \neq n$.

3.1 Relaxation

With the integral constraints on x_{im} being relaxed, the convex programming described in Equation (3) is rewritten as:

$$\begin{array}{ll} \text{minimize} & \sum_{\tau_i \in \mathbf{T}} E_i(t_i), \\ \text{subject to} & \sum_{\tau_i \in \mathbf{T}} t_i / p_i = M, \text{ and} \\ & 0 < t_i \leq p_i, \end{array} \quad (4)$$

where $E_i(t_i)$ is $\hat{E}_i(t_i)\frac{L}{p_i}$. Let $\bar{E}_i()$ be defined as $-E_i()$. The Karush-Kuhn-Tucker (KKT) optimality condition for Equation (4) is to find $(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathbf{T}|})$, $(t_1^*, t_2^*, \ldots, t_{|\mathbf{T}|}^*)$, and a constant λ such that

$$\left\{ \begin{array}{l} \left\{ \begin{array}{ll} \bar{E}_i'(t_i^*) - \lambda_i / p_i = \lambda / p_i, & t_i^* / p_i \leq 1, \\ (t_i^* / p_i - 1)\lambda_i = 0, & \lambda_i \geq 0, \end{array} \right. \forall \tau_i \in \mathbf{T}, \text{ and} \\ \qquad \sum_{\tau_i \in \mathbf{T}} t_i^* / p_i = M, \end{array} \right. \quad (5)$$

where $\bar{E}_i'()$ is the derivative of $\bar{E}_i()$.

If t_i^* is set as p_i for $i = 1, 2, \ldots, \ell$, Equation (4) can be relaxed as follows.

$$\begin{array}{ll} \text{maximize} & \sum_{i=\ell+1}^{|\mathbf{T}|} \bar{E}_i(t_i) \\ \text{subject to} & \sum_{i=\ell+1}^{|\mathbf{T}|} t_i / p_i = (M - \ell), \end{array} \quad (6)$$

by further ignoring the inequality $t_i \leq p_i$. Equation (6) can be solved by applying the Lagrange Multiplier method. Since $\bar{E}'_i(t_i) = \frac{L}{p_i}(\alpha - 1)h_i t_i^{-\alpha}$, given an index ℓ, the conditions $p_i\bar{E}'_i(t_i) = p_j\bar{E}'_j(t_j)$ hold for all $\ell < i, j \leq |\mathbf{T}|$ for the Lagrange Multiplier method. Therefore, the optimal solution for Equation (6) is to assign $(t_{\ell+1}, t_{\ell+2}, \ldots, t_{|\mathbf{T}|})$ as $(t^*_{\ell+1}, t^*_{\ell+2}, \ldots, t^*_{|\mathbf{T}|})$, where

$$\sum_{j=\ell+1}^{|\mathbf{T}|} \frac{t^*_{\ell+1}}{p_j} \sqrt[\alpha]{\frac{h_j}{h_{\ell+1}}} = (M - \ell), \tag{7a}$$

$$t^*_j = (t^*_{\ell+1}) \sqrt[\alpha]{\frac{h_j}{h_{\ell+1}}}, \forall \ell+1 < j \leq |\mathbf{T}|, \tag{7b}$$

and the Lagrange multiplier λ is $p_{\ell+1}\bar{E}'_{\ell+1}(t^*_{\ell+1})$. Let $\mathbf{T}$ be a sorted set by a *non-increasing* order of $p_i\bar{E}'_i(p_i)$. The following lemma helps obtain an optimal solution for Equation (4).

Lemma 1 *Suppose that each t^*_j in $(t^*_{\ell^*+1}, t^*_{\ell^*+2}, \ldots, t^*_{|\mathbf{T}|})$ obtained in Equation (7) is less than p_j for an index ℓ^* and that $p_\ell\bar{E}'_{\ell^*}(p_\ell)$ is no less than $p_{\ell+1}\bar{E}'_{\ell^*+1}(t^*_{\ell^*+1})$, where $1 \leq \ell^* < M$. Then, assigning t_i as p_i for $i = 1, 2, \ldots, \ell^*$ and t_j as t^*_j for $j = \ell^*+1, \ell^*+2, \ldots, |\mathbf{T}|$ leads to an optimal solution for Equation (4).*

Proof. It is proved by verifying that all conditions in Equation (5) hold when (1.) $\lambda = p_{\ell^*+1}\bar{E}'_{\ell^*+1}(t^*_{\ell^*+1})$, (2.) $\lambda_j = 0$, for $j = \ell^*+1, \ell^*+2, \ldots, |\mathbf{T}|$, and (3.) $\lambda_i = p_i\bar{E}'_i(p_i) - p_{\ell^*+1}\bar{E}'_{\ell^*+1}(t^*_{\ell^*+1})$, for $i = 1, 2, \ldots, \ell^*$. □

Therefore, the optimal solution for Equation (4) can be obtained in $O(M|\mathbf{T}| + |\mathbf{T}|\log|\mathbf{T}|)$ by setting ℓ sequentially. Moreover, it can be obtained in $O(|\mathbf{T}|\log|\mathbf{T}|)$ by a binary search of ℓ. Let $(t^*_1, t^*_2, \ldots, t^*_{|\mathbf{T}|})$ be an optimal solution for the programming in Equation (4). We have the following lemma.

Lemma 2 *When $t^*_i < p_i$ and $t^*_j < p_j$, $p_iE'_i(t^*_i) = p_jE'_j(t^*_j)$, where $E'_i()$ and $E'_j()$ are the derivatives of $E_i()$ and $E_j()$, respectively.*

Proof. The lemma comes from the fact that $E'_i(t^*_i) = \frac{-\lambda}{p_i}$ and $E'_j(t^*_j) = \frac{-\lambda}{p_j}$ for a constant λ when $t^*_i < p_i$ and $t^*_j < p_j$. □

Properties for the derived solution in the relaxation phase In addition to the optimality of the derived solutions in the relaxation phase, the solutions have some interesting properties, which will be specified later in this subsection and widely used in this paper. For the rest of this paper, let the utilization $u^*_i = t^*_i/p_i$ of task τ_i in $\mathbf{T}$ derived in this relaxation phase be defined as the *estimated utilization* of τ_i, and e^*_i be the *estimated expected 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 $\mathbf{T}$ consisting of the tasks whose estimated utilizations are 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 $\mathbf{T} \setminus \mathbf{T}'$. The following lemma concerning the relationship between two tasks in task set $\mathbf{T}'$ will be widely used in this paper.

Lemma 3 *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 $h_i\frac{L}{p_i}\frac{1}{(t^*_i)^\alpha} \cdot p_i = h_j\frac{L}{p_j}\frac{1}{(t^*_j)^\alpha} \cdot p_j$ in Lemma 2, we know that $\frac{u^*_i}{u^*_j} = \frac{e^*_i}{e^*_j}$. □

Algorithm 1 : LEUF

Input: $(\mathbf{T}, M)$;

if $|\mathbf{T}| \leq M$ **then**
  return the schedule to execute each task τ_i in $\mathbf{T}$ on processor i;
determine the optimal solution for Equation (4), and obtain u^*_i for every $\tau_i \in \mathbf{T}$;
sort $\mathbf{T}$ in a non-increasing order of their estimated utilizations;
$U_1 \leftarrow \cdots \leftarrow U_M \leftarrow 0$, and $\mathbf{T}_1 \leftarrow \cdots \leftarrow \mathbf{T}_M \leftarrow \emptyset$;
for $i \leftarrow 1$ to $|\mathbf{T}|$ **do**
  find the smallest U_m; (break ties by choosing the smallest index m)
  $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$ and $U_m \leftarrow U_m + u^*_i$;
return the schedule S_{LEUF} which executes task τ_i in $\mathbf{T}_m$ $(1 \leq m \leq M)$ on processor m;

Suppose that $\phi(\mathbf{T}^\dagger)$ is the minimum expected energy consumption in the hyper-period of the tasks in $\mathbf{T}$ to complete all the tasks in task set $\mathbf{T}^\dagger$ in time on a processor. By applying the KKT optimality condition, $\phi(\mathbf{T}^\dagger)$ is equal to $(\sum_{\tau_i \in \mathbf{T}^\dagger} e^*_i)(\sum_{\tau_i \in \mathbf{T}^\dagger} u^*_i)^{\alpha-1}$ when all the tasks in $\mathbf{T}^\dagger$ are in $\mathbf{T}'$. Hence, with Lemma 3, we have $\phi(\mathbf{T}^\dagger) = \frac{e^*_r}{u^*_r}(\sum_{\tau_i \in \mathbf{T}^\dagger} u^*_i)^\alpha$ for some task τ_r in $\mathbf{T}'$ when $\mathbf{T}^\dagger \subseteq \mathbf{T}'$.

Moreover, suppose that $\mathbf{T}^\dagger_m$ contains at least two tasks to be scheduled on processor m, the total estimated utilization of $\mathbf{T}^\dagger_m$ is no less than that of task set $\mathbf{T}^\dagger_n$, the total estimated utilization of task set $\mathbf{T}^\dagger_n$ on processor n is no more than 1, $\mathbf{T}^\dagger_n \subseteq \mathbf{T}'$, and $\sum_{\tau_\ell \in \mathbf{T}^\dagger_m} u^*_\ell > (\sum_{\tau_\ell \in \mathbf{T}^\dagger_n} u^*_\ell) + u^*_k$, where τ_k is the task with the smallest estimated expected energy consumption in $\mathbf{T}^\dagger_m$. We have the following inequality:

$$\phi(\mathbf{T}^\dagger_m) + \phi(\mathbf{T}^\dagger_n) > \phi(\mathbf{T}^\dagger_m \setminus \{\tau_k\}) + \phi(\mathbf{T}^\dagger_n \cup \{\tau_k\}), \tag{8}$$

which indicates the convexity of the estimated expected energy consumption.

Lemma 4 *There exists an optimal schedule that executes each task $\tau_i \in \hat{\mathbf{T}}$ entirely on an individual processor.*

Proof. It is proved by applying Equation (8) directly. □

3.2 An efficient 1.13-approximation algorithm

We derive a feasible schedule based on the estimated utilizations of tasks derived in the relaxation phase, i.e., $(u^*_1, u^*_2, \ldots, u^*_{|\mathbf{T}|})$, by adopting the *Largest-Estimated-Utilization-First* strategy. The proposed algorithm called LEUF is shown in Algorithm 1. Let $\mathbf{T}_m$ denote the set of the tasks assigned to processor m, which is an empty set initially. U_m denotes the *total estimated utilization* on processor m, which is defined as the sum of the estimated utilizations of the tasks in $\mathbf{T}_m$. 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 processor m 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 $\mathbf{T}_m$. That is, the execution time of every job of task τ_i is set as $\frac{t_i^*}{U_m}$. The computation requirement $X_{i,j}$ of task τ_i in $\mathbf{T}_m$ is executed at speed $U_m\left(\frac{\sum_{b=1}^{\beta_i} X_{i,b}\sqrt[\alpha]{\Psi_i^\dagger(b)}}{t_i^* \sqrt[\alpha]{\Psi_i^\dagger(j)}}\right)$. The time complexity of Algorithm LEUF is $O(|\mathbf{T}|\log|\mathbf{T}|)$. For simplicity of representation, any schedule derived from Algorithm LEUF is denoted by S_{LEUF}.

The load-balancing approach in [25] uses the value $\frac{\sum_{b=1}^{\beta_i} X_{i,b}\sqrt[\alpha]{\Psi_i^\dagger(b)}}{p_i}$, denoted by *variant load* here, as the load-balancing factor to assign task τ_i, where $X_{i,b}$ was assumed a constant and α was 3 in [25]. It inserts the un-assigned task with the largest variant load to the processor with the smallest summation of the variant loads of the assigned tasks on it so far until all the tasks are assigned. It is not difficult to see that it derives the same task partition as Algorithm LEUF does. Jensen's Inequality motivates the load-balancing approach adopted in [25], but there was no theoretical performance analysis yet. Here, by using the estimated utilization, we can show that the approach has performance guarantees.

To prove the approximation ratio of Algorithm LEUF, our strategy is to build a lower bound of the expected energy consumption of feasible schedules and an upper bound of the expected energy consumption of the schedule derived from Algorithm LEUF. The upper bound divided by the lower bound is the approximation ratio of Algorithm LEUF.

Lower bound of the expected energy consumption of feasible schedules We now derive a lower bound of the multiprocessor expected-energy-efficient scheduling problem for a given task set $\mathbf{T}$. The lower bound is based on a relaxation of the considered problem, in which some tasks might be executed simultaneously on more than one processor. Before presenting the lower bound, we first show that Algorithm LEUF derives optimal solutions for some special cases. In such a special case, Algorithm LEUF derives a task partition by assigning at most two tasks on a processor, in which moving any task on a processor to another or switching any two tasks on two processors does not decrease the expected energy consumption.

Lemma 5 *Algorithm* LEUF *derives an optimal solution for the multiprocessor expected-energy-efficient scheduling problem if* $|\mathbf{T}'| \leq 2(M - |\hat{\mathbf{T}}|)$ *and* $u_{i+M}^* \geq \frac{1}{2}u_{M-i+1}^*$ *for all* $1 \leq i \leq |\mathbf{T}| - M$.

Proof. Due to space limitations, the detailed proof is in a tech report [10]. □

Based on the optimality of Algorithm LEUF described in Lemma 5, we now derive a lower bound of the expected energy consumption of feasible schedules for any task set $\mathbf{T}$. Let k^* be the largest index k satisfying $M \leq k \leq 2(M - |\hat{\mathbf{T}}|)$ and $u_{i+M}^* \geq \frac{1}{2}u_{M-i+1}^*$ for all $1 \leq i \leq k - M$. $\mathbf{T}^f$ represents the set of the first k^* tasks of $\mathbf{T}$. We introduce another relaxed problem, referred to as the *semi-relaxed multiprocessor expected-energy-efficient scheduling* problem:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{\tau_i \in \mathbf{T}} E_i(t_i) \\ \text{subject to} \quad & \textstyle\sum_{\tau_i \in \mathbf{T}} x_{im} \cdot t_i/p_i = 1, \text{ for } \quad m = 1, \ldots, M \\ & \textstyle\sum_{m=1}^{M} x_{im} = 1, \quad \forall \tau_i \in \mathbf{T}, \\ & x_{im} \in \{0,1\}, \quad \forall \tau_i \in \mathbf{T}^f, m = 1, \ldots, M, \\ & x_{im} \geq 0, \quad \forall \tau_i \in \mathbf{T} \setminus \mathbf{T}^f, \text{ and } m = 1, \ldots, M. \end{aligned} \tag{9}$$

The optimal solution for the semi-relaxed multiprocessor expected-energy-efficient scheduling problem can be derived efficiently as follows. First of all, we execute Algorithm LEUF($\mathbf{T}^f, M$) to get an optimal partition on $\mathbf{T}^f$. For the rest of this subsection, let $U_1^\dagger, U_2^\dagger, \ldots, U_M^\dagger$ be the total estimated utilizations and $\mathbf{T}_1^\dagger, \mathbf{T}_2^\dagger, \ldots, \mathbf{T}_M^\dagger$ be the resulting task partition for $\mathbf{T}^f$. Based on Equation (8), we should assign those tasks in $\mathbf{T} \setminus \mathbf{T}^f$ to the processors with smaller total estimated utilizations as possible. Hence, we find the constant $U_{\min}^\dagger$ such that $\sum_{m=1}^{M}(U_{\min}^\dagger - U_m^\dagger)\delta_{U_{\min}^\dagger > U_m^\dagger} = \sum_{\tau_i \in \mathbf{T}\setminus\mathbf{T}^f} u_i^*$, where $\delta_{U_{\min}^\dagger > U_m^\dagger}$ is 1 when $U_{\min}^\dagger > U_m^\dagger$, and 0, otherwise. For each τ_i in $\mathbf{T}^f$ assigned to $\mathbf{T}_m^\dagger$, let x_{im} be 1 and $x_{im'} = 0$ for any $m' \neq m$, while t_i is set as $p_i \frac{u_i^*}{\max\{U_{\min}^\dagger, U_m^\dagger\}}$. For each τ_i in $\mathbf{T} \setminus \mathbf{T}^f$, t_i is set as $p_i \frac{u_i^*}{U_{\min}^\dagger}$. For processor m with $U_{\min}^\dagger > U_m^\dagger$, we assign $U_{\min}^\dagger - U_m^\dagger$ estimated utilization for tasks in $\mathbf{T} \setminus \mathbf{T}^f$. As a result, the expected energy consumption for $\hat{\mathbf{T}}$ is $\sum_{\tau_i \in \hat{\mathbf{T}}} e_i^*$, that for $\mathbf{T}^f \setminus \hat{\mathbf{T}}$ is $\sum_{\tau_i \in \mathbf{T}^f \setminus \hat{\mathbf{T}}, m:\tau_i \in \mathbf{T}_m^\dagger} e_i^*(\max\{U_{\min}^\dagger, U_m^\dagger\})^{\alpha-1}$, and that for $\mathbf{T} \setminus \mathbf{T}^f$ is $\sum_{\tau_i \in \mathbf{T}\setminus\mathbf{T}^f} e_i^*(U_{\min}^\dagger)^{\alpha-1}$. Therefore, the resulting expected energy consumption for $\mathbf{T}$ of the semi-relaxed multiprocessor expected-energy-efficient scheduling problem is

$$\begin{aligned} & \textstyle\sum_{\tau_i \in \mathbf{T}^f \setminus \hat{\mathbf{T}}, m:\tau_i \in \mathbf{T}_m^\dagger} e_i^*(\max\{U_{\min}^\dagger, U_m^\dagger\})^{\alpha-1} \\ & + \textstyle\sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{\tau_i \in \mathbf{T}\setminus\mathbf{T}^f} e_i^*(U_{\min}^\dagger)^{\alpha-1}. \end{aligned}$$

The above algorithm for the semi-relaxed multiprocessor expected-energy-efficient scheduling problem is called Algorithm G-LEUF.

Lemma 6 *Algorithm* G-LEUF *derives the minimum expected energy consumption for the semi-relaxed multiprocessor expected-energy-efficient scheduling problem.*

Proof. It can be proved with very similar arguments to Lemma 5. □

The expected energy consumption of the solution derived from Algorithm G-LEUF, therefore, is the lower bound of that of any feasible solution of the input instance.

The approximation ratio of Algorithm LEUF We have shown a lower bound of expected energy consumption of Algorithm LEUF. We now show the approximation ratio of Algorithm LEUF by dividing the upper bound of the expected energy consumption of the derived solution by the lower bound derived above. The following lemma shows that the difference of the total estimated utilizations on processors is bounded.

Lemma 7 *For processor* m *with* $U_m^\dagger \geq U_{\min}^\dagger$, $U_m^\dagger$ *is equal to* U_m *for Algorithm* LEUF. *For processors* m^* *and* m' *with* $U_{m'}^\dagger < U_{\min}^\dagger$, $U_{m^*}^\dagger < U_{\min}^\dagger$, *and* $U_{m^*} \geq U_{\min}^\dagger \geq U_{m'}$, U_{m^*} *is at most* $\frac{3}{2}U_{m'}$, *where* U_{m^*} *and* $U_{m'}$ *are the total estimated utilizations on processors* m^* *and* m' *after calling Algorithm* LEUF *on* $\mathbf{T}$, *respectively.*

Proof. For any processor m with $U_m^\dagger \geq U_{\min}^\dagger$, once we consider task τ_i in $\mathbf{T} \setminus \mathbf{T}^f$ in the loop from Step 6 to Step 8 in Algorithm LEUF, there must be another processor with smaller total estimated utilization. Hence, Algorithm LEUF never assigns any task in $\mathbf{T} \setminus \mathbf{T}^f$ to processor m. Namely, $U_m = U_m^\dagger$.

We now prove the second case. Suppose that τ_k is the last task inserted into $\mathbf{T}_{m^*}$ when we execute Algorithm LEUF for $\mathbf{T}$. By definitions, τ_k is in $\mathbf{T} \setminus \mathbf{T}^f$. Since τ_k is inserted into $\mathbf{T}_{m^*}$ instead of $\mathbf{T}_{m'}$, we also know that $U_{m^*} - u_k^* \le U_{m'}$. If $\mathbf{T}_{m^*} \setminus \{\tau_k\}$ has only one task, u_k^* is smaller than $\frac{1}{2}(U_{m^*} - u_k^*)$; otherwise, τ_k must be in $\mathbf{T}^f$. If $\mathbf{T}_{m^*} \setminus \{\tau_k\}$ has more than one task, u_k^* is no more than $\frac{1}{2}(U_{m^*} - u_k^*)$ because of the largest estimated utilization first strategy. Hence, $u_k^* \le \frac{1}{2}U_{m'}$. As a result, $U_{m^*} \le \frac{3}{2}U_{m'}$. □

The following lemma is required to show the approximation ratio of Algorithm LEUF.

Lemma 8 *Suppose $f(y) = k \cdot (3y)^\alpha + (\hat{M} - k)(2y)^\alpha$ for a positive number $\hat{M}$ and a non-negative number k, where $0 \le y, 0 \le k \le \hat{M}$, and $k \cdot 3y + (\hat{M} - k) \cdot 2y = \hat{M}$, then $f(y) \le \frac{(\alpha-1)^{\alpha-1}(3^\alpha - 2^\alpha)^\alpha}{(2\cdot 3^\alpha - 3\cdot 2^\alpha)^{\alpha-1}\alpha^\alpha}\hat{M}$.*

Proof. Due to space limitations, the detailed proof is in a tech report [10]. □

Based on the above lemmas, the approximation ratio of the algorithm can be proved as follows:

Theorem 1 *Algorithm* LEUF *is a* 1.13*-approximation algorithm for the multiprocessor expected-energy-efficient scheduling problem.*

Proof. By the optimality of Algorithm G-LEUF, we have

$$\begin{aligned}\Phi(S^*) \ge\ & \sum_{\tau_i \in \mathbf{T}^f \setminus \hat{\mathbf{T}}, m: \tau_i \in \mathbf{T}_m^\dagger} e_i^* (\max\{U_{\min}^\dagger, U_m^\dagger\})^{\alpha-1} \\ & + \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{\tau_i \in \mathbf{T} \setminus \mathbf{T}^f} e_i^* (U_{\min}^\dagger)^{\alpha-1},\end{aligned}$$

where S^* is an optimal schedule for $\mathbf{T}$. The expected energy consumption of the schedule S_{LEUF} derived is

$$\Phi(S_{\text{LEUF}}) = \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{\tau_i \in \mathbf{T} \setminus \hat{\mathbf{T}}, m: \tau_i \in \mathbf{T}_m} e_i^* (U_m)^{\alpha-1}. \quad (10)$$

Suppose that $\mathbf{M}^\dagger$ is the set of processors in which $U_m^\dagger = U_m$ for every m in $\mathbf{M}^\dagger$. Let τ_r be some task in $\mathbf{T}'$. By Lemma 3, we know that $\Phi(S^*) \ge \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{m \in \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha + (M - |\mathbf{M}^\dagger|)\frac{e_r^*}{u_r^*}(U_{\min}^\dagger)^\alpha$ as well as $\Phi(S_{\text{LEUF}}) = \sum_{\tau_i \in \hat{\mathbf{T}}} e_i^* + \sum_{m \in \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha + \sum_{m \notin \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha$. The approximation ratio $\mathcal{A}$ is

$$\begin{aligned}\mathcal{A} = \frac{\Phi(S_{\text{LEUF}})}{\Phi(S^*)} & \le \frac{\sum_{m \in \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha + \sum_{m \notin \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha}{\sum_{m \in \mathbf{M}^\dagger} \frac{e_r^*}{u_r^*}(U_m)^\alpha + (M - |\mathbf{M}^\dagger|)\frac{e_r^*}{u_r^*}(U_{\min}^\dagger)^\alpha} \\ & \le \frac{\sum_{m \notin \mathbf{M}^\dagger}(U_m)^\alpha}{(M - |\mathbf{M}^\dagger|)(U_{\min}^\dagger)^\alpha},\end{aligned} \quad (11)$$

where $\sum_{m \notin \mathbf{M}^\dagger} U_m = (M - |\mathbf{M}^\dagger|)U_{\min}^\dagger$. Suppose that $U_{\hat{m}}$ is the minimum total estimated utilization after calling Algorithm LEUF. Based on Lemma 7, we have $1.5U_{\hat{m}} \ge U_m$, for all $m \notin \mathbf{M}^\dagger$. Because of the convexity of the function $(U_m)^\alpha$ of U_m, the fact $1.5U_{\hat{m}} - U_m \ge 0$, and $U_m - U_{\hat{m}} \ge 0$, we have $(U_m)^\alpha \le \frac{1.5U_{\hat{m}} - U_m}{0.5U_{\hat{m}}}(U_{\hat{m}})^\alpha + \frac{U_m - U_{\hat{m}}}{0.5U_{\hat{m}}}(1.5U_{\hat{m}})^\alpha$, since $\frac{1.5U_{\hat{m}} - U_m}{0.5U_{\hat{m}}}(U_{\hat{m}}) + \frac{U_m - U_{\hat{m}}}{0.5U_{\hat{m}}}(1.5U_{\hat{m}})$ is equal to U_m. Hence,

$$\sum_{m \notin \mathbf{M}^\dagger}(U_m)^\alpha \le k \cdot (1.5U_{\hat{m}})^\alpha + (M - |\mathbf{M}^\dagger| - k)(U_{\hat{m}})^\alpha,$$

where $1.5k \cdot U_{\hat{m}} + (M - |\mathbf{M}^\dagger| - k)U_{\hat{m}} = (M - |\mathbf{M}^\dagger|)U_{\min}^\dagger$. By applying Lemma 8 after setting y as $0.5\frac{U_{\hat{m}}}{U_{\min}^\dagger}$ and $\hat{M}$ as $(M - |\mathbf{M}^\dagger|)$, we have $\sum_{m \notin \mathbf{M}^\dagger}(U_m)^\alpha \le \frac{(\alpha-1)^{\alpha-1}(3^\alpha - 2^\alpha)^\alpha}{(2\cdot 3^\alpha - 3\cdot 2^\alpha)^{\alpha-1}\alpha^\alpha}(M - |\mathbf{M}'|)(U_{\min}^\dagger)^\alpha$. As a result, $\mathcal{A} \le \frac{(\alpha-1)^{\alpha-1}(3^\alpha - 2^\alpha)^\alpha}{(2\cdot 3^\alpha - 3\cdot 2^\alpha)^{\alpha-1}\alpha^\alpha}$, and this theorem is proved by observing that $\alpha \le 3$. □

3.3 A polynomial-time approximation scheme

This subsection presents a polynomial-time approximation scheme (PTAS) for the multiprocessor expected-energy-efficient scheduling problem. By Lemma 4, we only have to focus on scheduling tasks in $\mathbf{T}'$ on $M - |\hat{\mathbf{T}}|$ processors. For the rest of this subsection, we only consider systems that execute each individual task τ_i in $\hat{\mathbf{T}}$ on processor $M - |\hat{\mathbf{T}}| + i$. For notational brevity, we denote $M - |\hat{\mathbf{T}}|$ by M'. Let $\mathbf{T}_m^o$ be the set of tasks assigned onto processor m in an optimal schedule. Clearly, $\mathbf{T}_m^o \cap \mathbf{T}_{m'}^o = \emptyset$ for any $m \ne m'$ and $\cup_{m=1}^{M'} \mathbf{T}_m^o = \mathbf{T}'$.

To build the PTAS, we first categorize the tasks in $\mathbf{T}'$ into small tasks and large tasks. The estimated utilizations of those large tasks are rounded down to proper values, and then a polynomial-time algorithm which derives an optimal solution based on the rounded estimated utilizations is used to partition the large tasks. At the end, the small tasks are assigned onto processors without increasing too much expected energy consumption. To present the PTAS, we will first derive optimal solutions of special cases with a fixed number of distinct estimated utilizations of tasks. Then, we will detail the categorization of the large and small tasks, the rounding of estimated utilizations, the use of the polynomial-time algorithm for special cases, and the assignment of large and small tasks. At the end, the approximation ratio and the time complexity of the PTAS will be presented.

A polynomial-time algorithm for the derivation of optimal solutions of special cases Here, we consider the special case when the number of distinct estimated utilizations of tasks in $\mathbf{T}'$ is fixed. The algorithm which derives an optimal solution for special cases will be used for tasks with rounded estimated utilizations. For such cases, these fixed estimated utilizations are denoted by $v_1^*, v_2^*, \ldots, v_\kappa^*$, where κ is the number of fixed estimated utilizations of tasks in $\mathbf{T}$. Without loss of generality, $v_1^* < v_2^* < \cdots < v_\kappa^*$. The number of tasks in $\mathbf{T}'$ with estimated utilization equal to v_i^* is denoted by n_i. By Equation (8) and the fact that $\sum_{i=1}^{\kappa} v_i^* \cdot n_i \le M'$, we have the following lemma.

Lemma 9 *For an optimal schedule, the number of tasks assigned on each processor is at most $\left\lceil \frac{1}{v_1^*} \right\rceil$.*

Proof. Suppose that the cardinality of the task set $\mathbf{T}_m^o$ is at least $\left\lceil \frac{1}{v_1^*} \right\rceil + 1$ and τ_k is the task with the minimum estimated utilization in $\mathbf{T}_m^o$. Clearly, we know that $\sum_{\tau_i \in \mathbf{T}_m^o} u_i^* \ge 1 + u_k^*$. By the pigeon-hole principle, we know that there must be a processor m' with $\sum_{\tau_i \in \mathbf{T}_{m'}^o} u_i^* < 1$. By Equation (8), we know that $\phi(\mathbf{T}_m^o) + \phi(\mathbf{T}_{m'}^o) > \phi(\mathbf{T}_m^o \setminus \{\tau_k\}) + \phi(\mathbf{T}_{m'}^o \cup \{\tau_k\})$, which contradicts the optimality of $\mathbf{T}_1^o, \mathbf{T}_2^o, \ldots, \mathbf{T}_{M'}^o$. □

A *configuration* for a processor is defined as a vector $\vec{w} = (w_1, w_2, \dots, w_\kappa)$, in which $w_i \in \{0, 1, \dots, n_i\}$ denotes the number of tasks with v_i^* estimated utilization in $\mathbf{T}'$. By Lemma 9, to derive an optimal solution, we only have to consider configurations $\vec{w}$ on processors with $\sum_{i=1}^{\kappa} w_i \leq \left\lceil \frac{1}{v_1^*} \right\rceil$. As a result, there are at most $Q = \binom{\left\lceil \frac{1}{v_1^*} \right\rceil + \kappa}{\kappa}$ different configurations on a processor for optimal solutions. For a feasible solution to assigning tasks in $\mathbf{T}'$ on M' processors, let $\vec{w}_m$ be the corresponding configuration on processor m, in which $w_{m,i}$ denotes the number of tasks with v_i^* estimated utilization in $\mathbf{T}'$ in configuration $\vec{w}_m$. As a result, we only have to consider configurations with $\sum_{m=1}^{M'} \sum_{i=1}^{\kappa} w_{m,i} = |\mathbf{T}'|$, $\sum_{i=1}^{\kappa} w_{m,i} \leq \left\lceil \frac{1}{v_1^*} \right\rceil$, and $\sum_{m=1}^{M'} w_{m,i} = n_i$ for $i = 1, 2, \dots, \kappa$.

For a configuration $\vec{w}_m$ on processor m, the minimum expected energy consumption on executing the corresponding task set consisting of $w_{m,i}$ tasks with estimated utilization v_i^* is equal to $\frac{e_r^*}{v_r^*}(\sum_{i=1}^{\kappa} w_{m,i} v_i^*)^\alpha$, which can be obtained in $O(\kappa)$. Since the number of configurations to achieve $\sum_{m=1}^{M'} \sum_{i=1}^{\kappa} w_{m,i} = |\mathbf{T}'|$ and $\sum_{i=1}^{\kappa} w_{m,i} \leq \left\lceil \frac{1}{v_1^*} \right\rceil$ is $\binom{M'+Q}{Q} = O((M')^Q)$, we can derive an optimal assignment by enumerating the different configurations for $\mathbf{T}'$ on the M' processors with $\sum_{m=1}^{M'} w_{m,i} = n_i$ for $i = 1, 2, \dots, \kappa$ and picking up the solution with the minimum expected energy consumption.

Another method is to solve an integer linear programming with a fixed number of variables. Suppose that $\vec{q}$ is a configuration with at most $\left\lceil \frac{1}{v_1^*} \right\rceil$ tasks, and spans the configuration set $\vec{\mathbf{Q}}$. q_i is the number of tasks with estimated utilization v_i^* in configuration $\vec{q}$. Clearly, there are Q elements in $\vec{\mathbf{Q}}$. Let $x_{\vec{q}}$ be an integral variable between 0 and M', and $\Omega(\vec{q})$ be the minimum expected energy consumption on executing the corresponding task set consisting of q_i tasks with estimated utilization v_i^* without violating the timing constraints. We can formulate the multiprocessor expected-energy-efficient scheduling of task set $\mathbf{T}'$ on M' processors as follows:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{\vec{q} \in \vec{\mathbf{Q}}} x_{\vec{q}} \cdot \Omega(\vec{q}) \\ \text{subject to} \quad & \textstyle\sum_{\vec{q} \in \vec{\mathbf{Q}}} x_{\vec{q}} \leq M' \\ & \textstyle\sum_{\vec{q} \in \vec{\mathbf{Q}}} x_{\vec{q}} \cdot q_i = n_i, \forall i = 1, 2, \dots, \kappa, \text{, and} \\ & x_{\vec{q}} \in \{0, 1, 2, \dots, M'\}, \forall \vec{q} \in \vec{\mathbf{Q}}. \end{aligned} \tag{12}$$

There are Q variables in Equation (12), whose optimal solution can be found by Lenstra's algorithm [22, §18.4] with time complexity exponential in the number of variables and polynomial in the size of the coefficient of the programming. Hence, an optimal solution for Equation (12) can be obtained in $O((2Q^3 2^{Q(Q-1)/4})^Q \log^{O(1)}(|\mathbf{T}'| + M'))$ time. Since $|\mathbf{T}'| > M'$ and Q is a constant when the number of different estimated utilizations in $\mathbf{T}'$ is fixed and the minimum estimated utilization in $\mathbf{T}'$ is a constant, the time complexity is $O(\log^{O(1)} |\mathbf{T}'|)$. The total time complexity to derive an optimal solution is $O(|\mathbf{T}| \log |\mathbf{T}| + \log^{O(1)} |\mathbf{T}'|)$. As a result, we have the following theorem.

Theorem 2 *A schedule with the minimum expected energy consumption for task set $\mathbf{T}'$ with $\frac{e_i^*}{u_i^*} = \frac{e_j^*}{u_j^*}$ for any two tasks $\tau_i, \tau_j \in \mathbf{T}'$ on M' processors can be solved in $O(|\mathbf{T}| \log |\mathbf{T}| + \log^{O(1)} |\mathbf{T}'|)$ when the number of different estimated utilizations in $\mathbf{T}'$ is fixed and the minimum estimated utilization in $\mathbf{T}'$ is a constant.*

The procedure of a polynomial-time approximation scheme for general cases Our polynomial-time approximation scheme for the multiprocessor expected-energy-efficient scheduling problem is based on rounding the estimated utilization on each task so that the number of different estimated utilizations and the number of different estimated expected energy consumptions are fixed. After rounding the parameters on tasks, we can then adopt Theorem 2 to derive an optimal schedule. A solution for scheduling tasks in $\mathbf{T}'$ on M' processors can then be determined, and can be shown energy-efficient.

Let ϵ be a fixed constant specified by users. We classify tasks in $\mathbf{T}'$ into two types. For any task τ_i in $\mathbf{T}'$ with $u_i^* \geq \epsilon$, we denote such a task as a *large* task. On the other hand, task τ_i in $\mathbf{T}'$ is referred to a *small* task if $u_i^* < \epsilon$. Let $\mathbf{B}_1$ ($\mathbf{B}_2$, respectively) be the task set which consists of large (small, respectively) tasks in $\mathbf{T}'$. For each large task τ_i in $\mathbf{B}_1$, let k_i be the integer with $\epsilon + k_i\epsilon^2 \leq u_i^* < \epsilon + (k_i + 1)\epsilon^2$. Since $u_i^* < 1$ for all $\tau_i \in \mathbf{T}'$, we know that $0 \leq k_i < \frac{1-\epsilon}{\epsilon^2}$. For each large task τ_i in $\mathbf{B}_1$, we create a rounded task $\tau_i^\flat$ by shrinking the estimated utilization $u_i^\flat$ as $(\epsilon + k_i\epsilon^2)$. Moreover, let the estimated expected energy consumption $e_i^\flat$ of rounded task $\tau_i^\flat$ be $e_i^* \frac{\epsilon + k_i\epsilon^2}{u_i^*}$. The constructed set of rounded tasks is denoted by $\mathbf{B}_1^\flat$. Hence, the number of distinct estimated utilizations in $\mathbf{B}_1^\flat$ is at most $\lceil \frac{1-\epsilon}{\epsilon^2} \rceil$.

We can derive an optimal solution on scheduling rounded tasks in $\mathbf{B}_1^\flat$ on M' processors in polynomial time by applying the algorithm presented in Theorem 2. Let $\mathbf{T}_m^\flat$ be the set of tasks assigned on processor m after applying the optimal algorithm for $\mathbf{B}_1^\flat$. For each rounded task $\tau_i^\flat$ in $\mathbf{T}_m^\flat$, we assign task τ_i to processor m. For notational brevity, let $U_m^\flat$ be the estimated utilization of rounded tasks in $\mathbf{T}_m^\flat$, i.e., $U_m^\flat = \sum_{\tau_i^\flat \in \mathbf{T}_m^\flat} u_i^\flat$.

Now, we show how to assign those small tasks in $\mathbf{B}_2$ onto processors. First, we find the minimum $U_{\min}^\flat$ such that $\sum_{m=1}^{M'} (U_{\min}^\flat - U_m^\flat)\delta_{U_{\min}^\flat > U_m^\flat} = \sum_{\tau_i \in \mathbf{B}_2} u_i^*$, where $\delta_{U_{\min}^\flat > U_m^\flat}$ is 1 when $U_{\min}^\flat > U_m^\flat$, and 0, otherwise. Let task set $\mathbf{B}_2'$ be a working task set, which is initialized as task set $\mathbf{B}_2$. Then, for each processor m with $m \leq M'$ and $U_{\min}^\flat > U_m^\flat$, we find a subset $\mathbf{B}_2^\dagger$ of task set $\mathbf{B}_2'$ to be assigned. If $\sum_{\tau_i \in \mathbf{B}_2'} u_i^* \leq U_{\min}^\flat - U_m^\flat$, let $\mathbf{B}_2^\dagger$ be $\mathbf{B}_2'$; otherwise, let $\mathbf{B}_2^\dagger$ be a subset of task set $\mathbf{B}_2'$ with $U_{\min}^\flat - U_m^\flat \leq \sum_{\tau_i \in \mathbf{B}_2^\dagger} u_i^* < \epsilon + U_{\min}^\flat - U_m^\flat$. We then assign all the tasks in $\mathbf{B}_2^\dagger$ on processor m and shrink the task set $\mathbf{B}_2'$ by subtracting $\mathbf{B}_2^\dagger$. For brevity, let $\mathbf{B}_{2,m}^\dagger$ be $\mathbf{B}_2^\dagger$.

Let the resulting task assignment be $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_{M'}$. The resulting schedule is returned. The pseudo-code of the algorithm, denoted by Algorithm ROUNDING, is in Algorithm 2.

Algorithm 2 :ROUNDING

Input: $\mathbf{T}, M, \epsilon$;

1: **if** $|\mathbf{T}| \leq M$ **then**
2: return the schedule to execute each task τ_i in $\mathbf{T}$ on processor i;
3: determine the optimal solution for Equation (4), and obtain u_i^* for every $\tau_i \in \mathbf{T}$;
4: $\mathbf{T}' \leftarrow \{\tau_i \mid \tau_i \in \mathbf{T}$ and $u_i^* < 1\}$, $\hat{\mathbf{T}} \leftarrow \mathbf{T} \setminus \mathbf{T}'$;
5: $M' \leftarrow M - |\hat{\mathbf{T}}|$;
6: schedule each task in $\mathbf{T} \setminus \mathbf{T}'$ on an individual processor;
7: $\mathbf{B}_1 \leftarrow \{\tau_i \mid \tau_i \in \mathbf{T}'$ and $u_i^* \geq \epsilon\}$, $\mathbf{B}_2 \leftarrow \mathbf{T}' \setminus \mathbf{B}_1$;
8: create a rounded task $\tau_i^\flat$ by setting its estimated utilization equal to $\epsilon + k_i\epsilon^2$ for each large task τ_i in $\mathbf{B}_1$, where $\epsilon + k_i\epsilon^2 \leq u_i^* < \epsilon + (k_i+1)\epsilon^2$;
9: $\mathbf{B}_1^\flat \leftarrow \{\tau_i^\flat \mid \tau_i \in \mathbf{B}_1\}$;
10: apply the algorithm in Theorem 2 to derive the task partition for task set $\mathbf{B}_1^\flat$ on M' processors, where $\mathbf{T}_m^\flat$ is the set of rounded tasks on processor m for the partition;
11: $U_m^\flat \leftarrow \sum_{\tau_i^\flat \in \mathbf{T}_m^\flat} u_i^\flat$ for $m = 1, 2, \ldots, M'$;
12: find $U_{\min}^\flat$ such that $\sum_{m=1}^{M'} (U_{\min}^\flat - U_m^\flat)\delta_{U_{\min}^\flat > U_m^\flat} = \sum_{\tau_i \in \mathbf{B}_2} u_i^*$, where $\delta_{U_{\min}^\flat > U_m^\flat}$ is 1 when $U_{\min}^\flat > U_m^\flat$, and 0, otherwise;
13: $\mathbf{B}_2' \leftarrow \mathbf{B}_2$;
14: **for** $m \leftarrow 1; m \leq M'; m \leftarrow m+1$ **do**
15: $\mathbf{T}_m \leftarrow \emptyset$;
16: for each rounded task $\tau_i^\flat$ in $\mathbf{T}_m^\flat$, $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$;
17: $\mathbf{B}_2^\dagger \leftarrow \emptyset$;
18: **if** $U_{\min}^\flat > U_m^\flat$ **then**
19: **for** each task τ_i in $\mathbf{B}_2'$ **do**
20: **if** $\sum_{\tau_i \in \mathbf{B}_2^\dagger} u_i^* < U_{\min}^\flat - U_m^\flat$ **then**
21: $\mathbf{B}_2^\dagger \leftarrow \mathbf{B}_2^\dagger \cup \{\tau_i\}$;
22: $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \mathbf{B}_2^\dagger$;
23: $\mathbf{B}_2' \leftarrow \mathbf{B}_2' \setminus \mathbf{B}_2^\dagger$, $\mathbf{B}_{2,m}^\dagger \leftarrow \mathbf{B}_2^\dagger$;
24: schedule each task in $\mathbf{T}_m$ on processor m;
25: return the resulting schedule;

An example for Algorithm ROUNDING We use the following example to show how Algorithm ROUNDING works. Suppose that we have fourteen tasks in $\mathbf{T}'$ and M' is 3. The estimated utilizations of these tasks are illustrated in Table 1. By taking ϵ as 0.1, five tasks, i.e., $\tau_{14}, \tau_{13}, \tau_{12}, \tau_{11}, \tau_{10}$, are defined as small tasks, and the other large tasks are rounded down to the closest $\epsilon + k_i\epsilon^2$ as shown in Table 1. For example, since $\epsilon + 7\epsilon^2 \leq 0.1786 < \epsilon + 8\epsilon^2$ when $\epsilon = 0.1$, the estimated utilization of task τ_9 is rounded down to 0.17. Then, we find the optimal task partition for the rounded tasks in $\{\tau_1^\flat, \tau_2^\flat, \tau_3^\flat, \tau_4^\flat, \tau_5^\flat, \tau_6^\flat, \tau_7^\flat, \tau_8^\flat, \tau_9^\flat\}$, in which rounded task set $\{\tau_1^\flat, \tau_2^\flat\}$ is set as $\mathbf{T}_1^\flat$, $\{\tau_3^\flat, \tau_5^\flat, \tau_6^\flat\}$ as $\mathbf{T}_2^\flat$, and $\{\tau_4^\flat, \tau_7^\flat, \tau_8^\flat, \tau_9^\flat\}$ as $\mathbf{T}_3^\flat$. As a result, $U_1^\flat$ is 0.92, $U_2^\flat$ is 0.92, and $U_3^\flat$ is 0.96. Then, tasks τ_1 and τ_2 are assigned to processor 1, tasks τ_3, τ_5, and τ_6 are to processor 2, and tasks τ_4, τ_7, τ_8, and τ_9 are to processor 3. Then, Algorithm ROUNDING starts to assign the small tasks, in which $U_{\min}^\flat$ is 0.983433. By applying the procedures between Step 13 and Step 23 in Algorithm 2 with consideration to small tasks from τ_{14} to τ_{10}, tasks $\tau_{14}, \tau_{13}, \tau_{12}, \tau_{11}$ are assigned to processor 1, and task τ_{10} is assigned to processor 2. The estimated utilizations of the task partition on these three processors are 1.0205, 0.9975, and 0.982. Hence, the expected energy consumption of the solution is $\frac{e_r^*}{u_r^*}(1.0205^3 + 0.9975^3 + 0.982^3) = 3.002254\frac{e_r^*}{u_r^*}$ when α is 3 and some task τ_r in task set $\mathbf{T}'$. By setting ϵ as 0.05, the results are also shown in Table 1, where the expected energy consumption is $3.00155\frac{e_r^*}{u_r^*}$.

The analysis of the feasibility and the approximation ratio of the PTAS The following lemma shows that the resulting task assignment assigns each task exactly to one processor, which implies the feasibility of the derived schedule.

Lemma 10 *$\cup_{m=1}^{M'} \mathbf{T}_m = \mathbf{T}'$ and $\mathbf{T}_m \cap \mathbf{T}_{m'} = \emptyset$ for any $m \neq m'$.*

Proof. Since each rounded task in $\mathbf{B}_1^\flat$ is assigned to one processor, each large task in $\mathbf{B}_1$ is assigned to one processor. We only focus on showing that each small task τ_i in $\mathbf{B}_2$ is assigned to one processor. Suppose that $\mathbf{B}_2'$ is not empty after all the processors are considered. Since $\mathbf{B}_2'$ is not empty, we know that $\sum_{\tau_i \in \mathbf{B}_2} u_i^* > \sum_{m=1}^{M'} (U_{\min}^\flat - U_m^\flat)\delta_{U_{\min}^\flat > U_m^\flat} = \sum_{\tau_i \in \mathbf{B}_2} u_i^*$. We reach the contradiction. □

The following lemma comes from the definition of the shrinking of those large tasks in $\mathbf{B}_1$.

Lemma 11 *$\frac{u_i^*}{u_i^\flat} \leq 1 + \epsilon$, for any large task $\tau_i^\flat \in \mathbf{B}_1^\flat$, which is constructed from task τ_i in $\mathbf{T}'$.*

Proof. Recall that k_i is the integer with $\epsilon + k_i\epsilon^2 \leq u_i^* < \epsilon + (k_i+1)\epsilon^2$ and $\tau_i^\flat$ is defined as $\epsilon + k_i\epsilon^2$. We know that $u_i^* - u_i^\flat \leq \epsilon^2$. Hence, $\frac{u_i^*}{u_i^\flat} = 1 + \frac{u_i^* - u_i^\flat}{u_i^\flat} \leq 1 + \frac{\epsilon^2}{u_i^\flat} \leq 1 + \epsilon$, where the second inequality holds because of $u_i^\flat \geq \epsilon$. □

We need the following lemma to prove the approximation ratio.

Lemma 12 *$\sum_{i=1}^{m} (y_i + z)^\alpha \leq (\sqrt[\alpha]{m}z + \sqrt[\alpha]{\sum_{i=1}^{m}(y_i)^\alpha})^\alpha$, for any non-negative real numbers $y_1, y_2, \ldots, y_m$, any positive real number $\alpha \geq 1$, and positive integer m.*

Proof. Due to space limitations, the detailed proof is in a tech report [10]. □

The minimum expected energy consumption $\phi^\flat(\mathbf{T}_m^\flat)$ of task set $\mathbf{T}_m^\flat$ is $\frac{e_r^*}{u_r^*}(\sum_{\tau_i \in \mathbf{T}_m^\flat} u_i^\flat)^\alpha$ for some task $\tau_r \in \mathbf{T}'$. For notational brevity, let U_m^* be $U_{\min}^\flat$ if $U_{\min}^\flat > U_m^\flat$ and $U_m^\flat$ otherwise, i.e.,

$$U_m^* = \begin{cases} U_{\min}^\flat, & \text{if } U_{\min}^\flat > U_m^\flat, \\ U_m^\flat & \text{otherwise.} \end{cases}$$

The following lemma shows the relationship of the expected energy consumption of the rounded input instance to that of the optimal schedule for $\mathbf{T}'$ on M' processors, where $\mathbf{T}_m^o$ denotes the set of tasks assigned on processor m in the optimal schedule.

Lemma 13 *$\sum_{m=1}^{M'} \frac{e_r^*}{u_r^*}(U_m^*)^\alpha \leq \sum_{m=1}^{M'} \phi(\mathbf{T}_m^o)$, for some task $\tau_r \in \mathbf{T}'$.*

	τ_{14}	τ_{13}	τ_{12}	τ_{11}	τ_{10}	τ_9	τ_8	τ_7	τ_6	τ_5	τ_4	τ_3	τ_2	τ_1
u_i^*	0.0022	0.0084	0.0155	0.0591	0.0651	0.1786	0.1798	0.1923	0.2316	0.2353	0.4313	0.4655	0.4666	0.4687
$u_i^\flat$ ($\epsilon = 0.1$)	small	small	small	small	small	0.1700	0.1700	0.1900	0.2300	0.2300	0.4300	0.4600	0.4600	0.4600
processor	1	1	1	1	2	3	3	3	2	2	3	2	1	1
$u_i^\flat$ ($\epsilon = 0.05$)	small	small	small	0.0575	0.0650	0.1775	0.1775	0.1900	0.2300	0.2350	0.4300	0.4650	0.4650	0.4675
processor	1	1	2	1	2	3	3	3	2	2	3	2	1	1

Table 1. The estimated utilization and rounded estimated utilization of tasks in the example.

Proof. The proof is similar to the optimal solution for the semi-relaxed multiprocessor expected-energy-efficient scheduling problem in Section 3.2. □

The following lemma shows the ratio of the expected energy consumption of the derived schedule to that of the optimal schedule for $\mathbf{T}'$ on M' processors.

Lemma 14

$$\sum_{m=1}^{M'} \phi(\mathbf{T}_m) \leq (1+2\epsilon)^\alpha \sum_{m=1}^{M'} \phi(\mathbf{T}_m^o).$$

Proof. From Lemma 11, we know that $\sum_{\tau_i \in \mathbf{T}_m \cap \mathbf{B}_1} u_i^* \leq (1+\epsilon) \sum_{\tau_i^\flat \in \mathbf{T}_m^\flat} u_i^\flat = (1+\epsilon) U_m^\flat$. By the definition of $\mathbf{B}_{2,m}^\dagger$ and the fact that each task in $\mathbf{B}_2$ is with estimated utilization smaller than ϵ, we have $\sum_{\mathbf{B}_{2,m}^\dagger} u_i^* \leq (U_m^* - U_m^\flat) + \epsilon$. Combining the two inequalities, we know $\sum_{\tau_i \in \mathbf{T}_m} u_i^* \leq (1+\epsilon) U_m^* + \epsilon$. Hence,

$$\begin{aligned}
\sum_{m=1}^{M'} \phi(\mathbf{T}_m) &= \frac{e_r^*}{u_r^*} \sum_{m=1}^{M'} (\sum_{\tau_i \in \mathbf{T}_m} u_i^*)^\alpha \\
&\leq \frac{e_r^*}{u_r^*} \sum_{m=1}^{M'} ((1+\epsilon) U_m^* + \epsilon)^\alpha \\
&\leq^1 \frac{e_r^*}{u_r^*} \left(\sqrt[\alpha]{M'} \epsilon + (1+\epsilon) \sqrt[\alpha]{\sum_{m=1}^{M'} (U_m^*)^\alpha} \right)^\alpha \\
&\leq^2 \left(\epsilon (\sum_{m=1}^{M'} \phi(\mathbf{T}_m^o))^{\frac{1}{\alpha}} + (1+\epsilon) (\sum_{m=1}^{M'} \phi(\mathbf{T}_m^o))^{\frac{1}{\alpha}} \right)^\alpha \\
&= (1+2\epsilon)^\alpha \sum_{m=1}^{M'} \phi(\mathbf{T}_m^o),
\end{aligned}$$

where $\leq^1$ comes from Lemma 12 and $\leq^2$ comes from Lemma 13 with the fact that $\frac{e_r^*}{u_r^*} M' \leq \sum_{m=1}^{M'} \phi(\mathbf{T}_m^o)$. □

By taking ϵ with $(1+2\epsilon)^\alpha \leq (1+\zeta)$, we could reach the following theorem since α is a constant.

Theorem 3 *A schedule with $(1+\zeta) E_{opt}$ for any task set $\mathbf{T}$ can be solved in $O(|\mathbf{T}| \log |\mathbf{T}| + g(\frac{1}{\zeta}) \log^{O(1)} |\mathbf{T}|)$, where ζ is an user-input instance, $g(\frac{1}{\zeta})$ is a function of $\frac{1}{\zeta}$, and E_{opt} is the minimum expected energy consumption for $\mathbf{T}$ to complete all the tasks in $\mathbf{T}$ before their deadlines.*

3.4 Remarks

Deriving a feasible task partition for the multiprocessor expected-energy-efficient scheduling problem is $\mathcal{NP}$-complete if there is a speed constraint [6]. The problem is equivalent to the programming with one additional constraint in Equation (3) with $t_i \geq c_i / s_{\max}$, where $s_{\max}$ is the maximum speed of the system. Task rejection [9] or resource augmentation with constraint violation [7, 17] to violate the constraint on the maximum speed for a little might be needed. For resource augmentation, the relaxation phase must be revised, and the estimated utilization can be used to derive task partitions. After task partition is done, the task set on processor m is scheduled with the approach in [24] by augmenting the maximum speed as $s_{\max} \times \max\{1, U_m\}$, where U_m is the total estimated utilization of the task set.

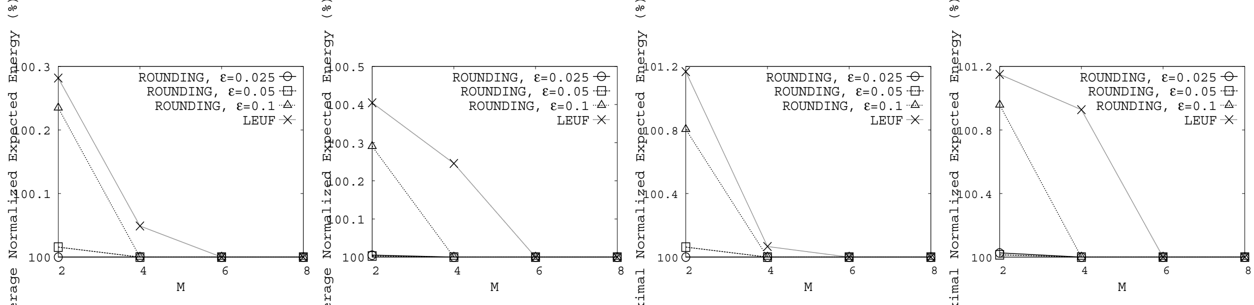
4 Performance Evaluation

This section provides performance evaluation of Algorithm LEUF and Algorithm ROUNDING, where the user-input parameter ϵ is set as 0.1, 0.05, and 0.025.

We perform simulations that apply the algorithms to a series of synthetic task sets with similar settings in [25]. Each task set consists of 8, 12, or 16 periodic tasks. The period p_i of each task τ_i ranges from 10 millisecond to 10 second. The number of the worst-case execution cycles c_i is between 100,000 and 500,000,000. Let β_i of each task τ_i be a random variable in the range of $[5, 20]$. The worst-case execution cycles c_i is evenly divided into β_i bins, i.e., $X_{i,b} = \lceil \frac{c_i}{\beta_i} \rceil$ for $b = 1, 2, \ldots, \beta_i$, while c_i is revised to $\beta_i X_{i,b}$ after the derivation of $X_{i,b}$. Then, we have to determine the probability density function $\psi_i()$ for each task τ_i. A Gaussian distribution and an exponential distribution are adopted here to determine the probability density function of each task. For task sets that are determined by a Gaussian distribution, the actual execution cycles of a task τ_i follows a Gaussian distribution in $(0, c_i]$, where the mean μ_i is chosen in $(0, c_i]$ and the standard deviation σ_i is $\frac{c_i}{6}$. For exponential distributed task sets, the actual execution cycles of a task τ_i is exponentially distributed in $(0, c_i]$, where $\frac{1}{\eta_i}$ is a random variable in the range of $(0, c_i]$, and $\mu_i = \frac{1}{\eta_i}$ and $\sigma_i^2 = \frac{1}{\eta_i^2}$.

We simulate each task set on platforms with 2, 4, 6, or 8 processors to evaluate the effect of the number of processors on the evaluated algorithms, while the value α is set as 3. The *normalized expected energy* is used to evaluate the effectiveness of the algorithms. The normalized expected energy of an algorithm for an input instance is the energy consumption of the schedule derived from the algorithm in the hyper-period of the task set divided by the energy of an optimal solution derived from an exhaustive search. Each configuration is done by simulating 64 task sets independently.[1] The average normalized expected energy and the maximal normalized expected energy are reported as the experimental results.

[1] Since the exhaustive search is very time-consuming, we are not able to have experiments with larger tasks, processors, and configurations.



(a) Gaussian distribution (b) Exponential distribution (c) Gaussian distribution (d) Exponential distribution

Figure 2. Experimental results when $|\mathbf{T}| = 8$: (a) and (b) for the average normalized expected energy and (c) and (d) for the maximal normalized expected energy.

Figure 2 shows the experimental results when there are 8 tasks for 2, 4, 6, or 8 processors. When $M = 6$ and $M = 8$, all the evaluated algorithms derive optimal solutions because most of the generated task sets satisfy the optimality condition stated in Lemma 5. However, when $M = 2$ and $M = 4$, Algorithm ROUNDING outperforms Algorithm LEUF no matter ϵ is set as 0.1, 0.05, or 0.025. Moreover, when $M = 2$ in Figure 2(d), setting ϵ as 0.05 has better results than setting ϵ as 0.025 does. This is because the setting of ϵ only reflects the worst-case guarantee, but the actual resulting schedule might be better if ϵ has a larger value. Figure 3 and Figure 4 are the experimental results when there are 12 and 16 tasks in $\mathbf{T}$, respectively. In both Figure 3 and Figure 4, Algorithm LEUF outperforms Algorithm ROUNDING in most cases when $\epsilon = 0.1$. But by setting ϵ as 0.025, Algorithm ROUNDING can perform better than Algorithm LEUF does in most cases.

5 Conclusion

This paper explores task partition and scheduling for the minimization of expected energy consumption in homogeneous multiprocessor systems with the capability of dynamic voltage scaling. The objective is to minimize the expected energy consumption for completing all the given tasks in time. By modeling the dynamic (or speed-dependent) power consumption function as s^α, we show that the Largest-Estimated-Utilization-First (LEUF) strategy is a $\frac{(\alpha-1)^{\alpha-1}(3^\alpha-2^\alpha)^\alpha}{(2\cdot 3^\alpha-3\cdot 2^\alpha)^{\alpha-1}\alpha^\alpha}$-approximation algorithm with $O(|\mathbf{T}|\log|\mathbf{T}|)$ time complexity, where s is the processor speed, α is a hardware-dependent factor between 1 and 3, and $\mathbf{T}$ is the set of the given real-time tasks. Since α is at most 3, the approximation ratio is at most 1.13. Moreover, with the rounding of the estimated worst-case utilization of tasks, we derive a polynomial-time approximation scheme (PTAS) to provide a $(1 + \zeta)$-approximated solution for any $1 > \zeta > 0$ for such a strongly $\mathcal{NP}$-hard problem. The proposed polynomial-time approximation scheme allows a system designer to trade the optimality of the derived solution with the analysis time.

An interesting extension for expected-energy-efficient scheduling is to consider the reduction of leakage power consumption by turning a processor off. In such a case, all the processors might not be used for task execution [7, 28]. However, to our best knowledge, the expected-energy-efficient scheduling with leakage-power consideration is still open even for uniprocessor systems. The task re-assignment approach [7] might be applied if the uniprocessor scheduling issue is resolved. It is also interesting to consider expected-energy-efficient scheduling on heterogeneous processors.

Acknowledgments We would like to thank Sathish Gopalakrishnan for his comments on the draft of this paper.

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 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.

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

[5] J.-J. Chen. Expected energy consumption minimization in dvs systems with discrete frequencies. In *ACM Symposium on Applied Computing*, 2008.

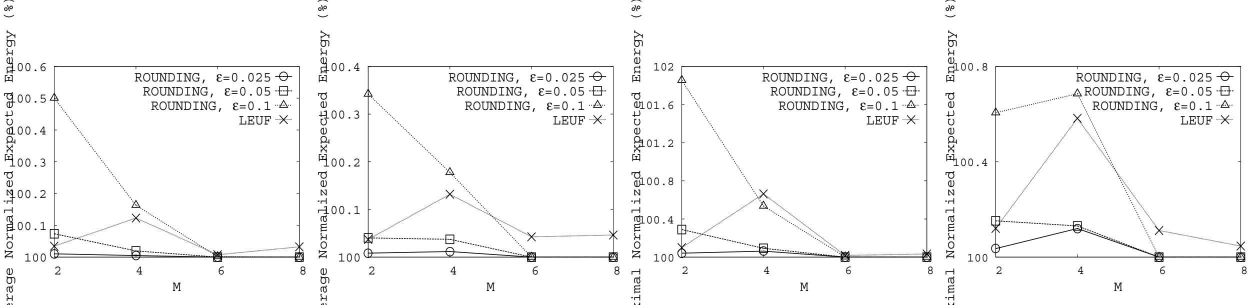
[6] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, and T.-W. Kuo. Multiprocessor energy-efficient scheduling with task migration considerations. In *EuroMicro Conference on Real-Time Systems (ECRTS'04)*, pages 101–108, 2004.

[7] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *IEEE Real-time and Embedded Technology and Applications Symposium*, pages 408–417, 2006.

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

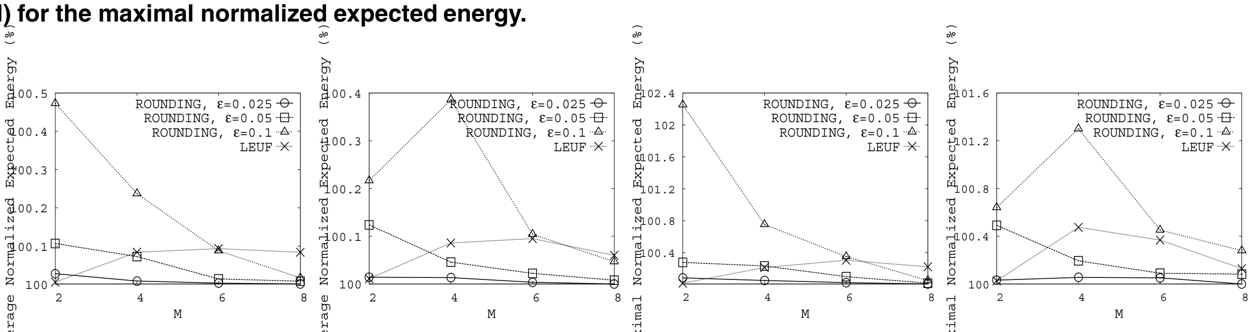
[9] J.-J. Chen, T.-W. Kuo, C.-L. Yang, and K.-J. King. Energy-efficient real-time task scheduling with task rejection. In *DATE*, pages 1629–1634, 2007.

[10] J.-J. Chen, C.-Y. Yang, T.-W. Kuo, and H.-I. Lu. Approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time. Technical report, NTU/NEWS-8-0001, National Taiwan University, 2008.



(a) Gaussian distribution (b) Exponential distribution (c) Gaussian distribution (d) Exponential distribution

Figure 3. Experimental results when $|\mathbf{T}| = 12$: (a) and (b) for the average normalized expected energy and (c) and (d) for the maximal normalized expected energy.



(a) Gaussian distribution (b) Exponential distribution (c) Gaussian distribution (d) Exponential distribution

Figure 4. Experimental results when $|\mathbf{T}| = 16$: (a) and (b) for the average normalized expected energy and (c) and (d) for the maximal normalized expected energy.

[11] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. In *ISLPED*, pages 46–51, 2001.

[12] 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.

[13] F. Gruian and K. Kuchcinski. Uncertainty-based scheduling: energy-efficient ordering for tasks with variable execution time. In *ISLPED*, pages 465–468, 2003.

[14] T.-Y. Huang, Y.-C. Tsai, and E. T.-H. Chu. A near-optimal solution for the heterogeneous multi-processor single-level voltage setup problem. In *21th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, 2007.

[15] C.-M. Hung, J.-J. Chen, and T.-W. Kuo. Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element. In *the 27th IEEE Real-Time Systems Symposium (RTSS)*, pages 303–312, 2006.

[16] 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, 2003.

[17] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Symposium on Theory of Computing*, pages 771–782. ACM Press, 1992.

[18] J. R. Lorch and A. J. Smith. Pace: A new approach to dynamic voltage scaling. *IEEE Trans. Computers*, 53(7):856–869, 2004.

[19] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron. Procrastinating voltage scheduling with discrete frequency sets. In *Design, Automation and Test in Europe (DATE)*, pages 456–461, 2006.

[20] 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.

[21] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice Hall, 2nd edition, 2002.

[22] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

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

[24] C. Xian and Y.-H. Lu. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. In *ACM Great Lakes Symposium on VLSI*, pages 392–397, 2006.

[25] C. Xian, Y.-H. Lu, and Z. Li. Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time. In *DAC*, pages 664–669, 2007.

[26] R. Xu, R. Melhem, and D. Mossé. A unified practical approach to stochastic dvs scheduling. In *The 7th ACM International Conference on Embedded Systems (EMSOFT)*, 2007.

[27] R. Xu, D. Mossé, and R. G. Melhem. Minimizing expected energy in real-time embedded systems. In *EMSOFT*, pages 251–254, 2005.

[28] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé. Energy-efficient policies for embedded clusters. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems(LCTES)*, pages 1–10, 2005.

[29] 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.

[30] 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.

[31] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *SOSP*, pages 149–163, 2003.

[32] 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.

[33] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal procrastinating voltage scheduling for hard real-time systems. In *DAC*, pages 905–908, 2005.