Multiprocessor Energy-Efficient Scheduling with Task Migration Considerations[*]

Jian-Jia Chen, Heng-Ruey Hsu, Kai-Hsiang Chuang,
Chia-Lin Yang, Ai-Chun Pang, and Tei-Wei Kuo
Department of Computer Science and Information
Engineering National Taiwan University, Taipei, Taiwan 106, ROC.
E-Mails: {r90079, b89108, b89109, yangc, acpang, ktw}@csie.ntu.edu.tw

Abstract

This paper targets energy-efficient scheduling of tasks over multiple processors, where tasks share a common deadline. Distinct from many research results on heuristics-based energy-efficient scheduling, we propose approximation algorithms with different approximation bounds for processors with/without constraints on the maximum processor speed, where no task migration is allowed. When there is no constraint on processor speeds, we propose an approximation algorithm for two-processor scheduling to provide trade-offs among the specified error, the running time, the approximation ratio, and the memory space complexity. An approximation algorithm with a 1.13-approximation ratio for M-processor systems is also derived ($M > 2$). When there is an upper bound on processor speeds, an artificial-bound approach is taken to minimize the energy consumption with a 1.13-approximation ratio. An optimal scheduling algorithm is then proposed in the minimization of the energy consumption when task migration is allowed.

Keywords: Energy-Efficient Scheduling, Real-Time Task Scheduling, Power Management, Real-Time Systems, Multiprocessor Scheduling.

1. Introduction

While an energy-efficient design has become a focus on various systems, voltage-scaling CPU's and power-aware subsystems are now adopted in many modern computer systems. The design of CPU circuitry is usually done such that a higher supply voltage results in a higher execution speed (or higher frequency). An example energy consumption function [1, 14], as follows, shows the energy consumption of a processor as a function of the processor speed:

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

where $s = k\frac{(V_{dd}-V_t)^2}{V_{dd}}$, and $P, s, C_{ef}, V_t, V_{dd}$, and k denote the energy consumption, the processor speed, 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 energy consumption function of a processor is usually a convex function of the processor speed, and each specific function is highly dependent on the design of the corresponding processor.[1]

Energy-efficient scheduling has been an active research topic in the past decade. In particular, Yao, et al. [15] proposed an off-line scheduling algorithm and an on-line competitive algorithm to minimize the energy consumption of task executions in a uniprocessor environment, where the processor under considerations has an infinite number of continuous processor speeds. In [10], Ishihara and Yasuura showed that an optimal schedule in the minimization of energy consumption with only two processor speeds when the processor has only a finite number of discrete processor speeds, and all tasks are ready at time 0 and have a common deadline. Note that the results could only be applied to processors with an energy consumption function equal to Formula (1). While an energy consumption function could be any convex function, Chen, et al. [3] showed that the result in [10] remains.

Although many excellent results were proposed for uniprocessor energy-efficient scheduling, little work has been done for multiprocessor envi-

* Support in parts by research grants from ROC National Science Council (NSC-92-2213-E-002-091, NSC-92-2213-E-002-092, and NSC-92-2220-E-002-013).

1 $f(x)$ is a convex function if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for any $\alpha \in (0,1)$ and any x, y [5].

ronments. In recent years, energy-efficient design has been outlined as a critical issue by the industry in business operations, e.g., [2], where various configurations of server farms are adopted. Unfortunately, multiprocessor energy-efficient scheduling is often NP-hard under various application constraints. Gruian [7] proposed a simulated annealing (SA) approach in multiprocessor energy-efficient scheduling with the considerations of precedence constraints and a predictable execution time for each task. In [8], a power-aware scheduling algorithm based on a list heuristics with a dynamic priority assignment was proposed to determine the amount of time allocated to each task. Zhang, et al. [16] proposed a heuristic algorithm in which each task was first assigned to a proper processor, and the processor speed in executing each task was then chosen without violating the precedence and timing constraints. Mishra, et al. [12] explored scheduling issues on the communication delay of tasks. Zhu, et al. [17] explored on-line scheduling for a set of independent/dependent frame-based tasks, where all tasks in a *frame-based task set* are ready at time 0 and share a common deadline. Given an off-line schedule with worst-case task execution times, on-line strategies were proposed to reclaim the slacks resulted from the early completion times of tasks observed in the run time. Although some work has been done on multiprocessor energy-efficient scheduling, many previous results are mainly on heuristics-based energy-efficient scheduling.

Distinct from the past work, the objective of this paper is to propose approximation algorithms with different approximation bounds for processors with/without constraints on the maximum processor speed, where task migration is considered. We first show that there does not exist any polynomial-time approximation algorithm with an approximation bound $(1 + \epsilon)$ in the minimization of energy consumption for multiprocessor scheduling over processors with an upper bound on the processor speed, where ϵ could be any positive real. When there is no constraint on processor speeds, we propose an approximation algorithm for two-processor scheduling to provide trade-offs among the specified error, the running time, the approximation ratio, and the memory space complexity. An approximation algorithm with a 1.13-approximation ratio for M-processor systems is also derived ($M > 2$). When there is an upper bound on processor speeds, an artificial-bound approach is taken to minimize the energy consumption with a 1.13-approximation ratio.[2] An optimal scheduling algorithm is then proposed in the minimization of the energy consumption when task migration is allowed.

The rest of this paper is organized as follows: Section 2 formally defines the multiprocessor energy-efficient scheduling problems and show their hardness. Section 3 presents approximation algorithms for multiprocessor energy-efficient scheduling for the two-processor case and general cases, where no task migration is allowed. In Section 4, an optimal polynomial-time scheduling algorithm is proposed in the minimization of the energy consumption when task migration is allowed. Section 5 is the conclusion.

2. Problem Definitions and NP-Hardness

2.1. Problem Definitions

This paper is interested in multiprocessor energy-efficient scheduling with/without constraints on the maximum processor speed and with task migration considerations. We assume a homogeneous multiprocessor environment, where each of the M identical processors has the same energy consumption function $P(s)$ of a given processor speed s. In this paper, $P(s)$ is assumed being a convex and increasing function. An example energy consumption function in [1, 14] is $P(s) = C_{ef} s((\frac{s}{k} + 2V_t)\frac{2V_t + \frac{s}{k} + \sqrt{\frac{4V_t s}{k} + \frac{s^2}{k^2}}}{2} - V_t^2)$, which is a reformulation of Formula (1). Let U_{max} denote the maximum available processor speed for processors under considerations such that tasks could be executed at any processor speed in $[0, U_{max}]$. When $V_t = 0$, $P(s) = \alpha s^3$, where $\alpha = \frac{C_{ef}}{k^2}$. It is reasonable to consider only cases for a given energy consumption function $P(s)$ where $P(s_1) > P(s_2)$ for $s_1 > s_2$. Let the energy consumed for a processor in the execution of tasks at the processor speed s for t time units be $P(s)t$. We assume that the number of CPU cycles executed in a time interval is linearly proportional to the processor speed. We denote the amount of required CPU cycles for a task running at a speed s for t time units is the multiplication of s and t.

Task migration might or might not be allowed in the exploring of energy-efficient scheduling in this paper. When task migration is allowed, migration cost is assumed being negligible. No task could execute simultaneously on more than one processors. For the rest of Section 2, we first formally define the multiprocessor scheduling problems with the minimization of en-

2 Such a constraint violation study was first introduced in [11].

ergy consumption with/without task migration in this paper. We then show the NP-hardness of the problems.

Definition 1 *Multiprocessor Scheduling with the Minimization of Energy Consumption with Task Migration* (MMEM)*:*

Consider a set T of independent tasks over M identical processors with an energy consumption function $P(s)$, where all tasks in T are ready at time 0 and share a common deadline D. Each task $\tau_i \in T$ is associated with a computation requirement equal to c_i CPU-cycles. The problem is to minimize the energy consumption in the scheduling of tasks in T without missing the common deadline D, where task migration is allowed.

A variation of the MMEM problem without task migration could be defined similarly as follows:

Definition 2 *Multiprocessor Scheduling with the Minimization of Energy Consumption without Task Migration* (MME)*:*

The input and output of the MME *problem are as the same as their counterparts of the* MMEM *problem, where no task migration is allowed.*

A *schedule* of a task set is a mapping of the executions of the tasks in the set to processors in the system with an assignment of processor speeds for the corresponding time intervals of the tasks. A schedule is *feasible* if all processor speeds assigned for its time intervals are valid, no task misses the deadline D, and the given task migration constraint is satisfied. The energy consumption of a schedule SC is denoted as $\Phi(SC)$ (Please see the first paragraph of this section for the definition of the energy consumption). A schedule is optimal if it is feasible, and its energy consumption is equal to the minimum energy consumption of all feasible schedules. If there does not exist any feasible schedule for an input instance, then the minimum energy consumption is denoted as ∞.

2.2. Hardness of the MME Problem

We shall show the NP-hardness of the MME problem in this section and then propose an optimal algorithm for the MMEM problem in a later section:

Lemma 1 (Chen, Kuo, and Yang [3]) *There exists an optimal schedule for any task set T executing on a single processor at the single processor speed $\frac{\sum_{\tau_i \in T} c_i}{D}$, where the processor under considerations has an infinite number of continuous processor speeds, and all tasks in T are ready at time 0 and have a common deadline D.*

Although multiprocessor scheduling is NP-hard [4] when no task migration is allowed, this does not imply the NP-hardness of the MME problem directly . For example, when $P()$ is a linear function($P(s) \propto s$), any feasible schedule is an optimal solution.

Theorem 1 *The* MME *problem is NP-hard when $M \geq 2$.*

Proof: The NP-hardness is proved by a reduction from the 3-PARTITION problem [4], where $P()$ is a strict convex and increasing function, follows from Lemma 1.[3] ■

A polynomial-time $(1+\epsilon)$-approximation algorithm must have a polynomial-time complexity of the input size and derive a solution with a bound $(1+\epsilon)$ on the given objective function [13]. That is, when $E(OPT)$ represents the value of an optimal solution for the objective function, any solution derived from a $(1+\epsilon)$-approximation algorithm should have a value of the objective function no more than $(1+\epsilon)E(OPT)$ (for minimization problems).

Theorem 2 *There does not exist a polynomial-time $(1+\epsilon)-$approximation algorithm for the* MME *problem ($\epsilon > 0$) when $U_{max} \neq \infty$, unless $P = NP$.*

Proof: This theorem can be proved by contradiction: Suppose that there exists a polynomial-time $(1+\epsilon)-$approximation algorithm for the MME problem, called *ALG*. Given an instance of the PARTITION problem [4] (which is NP-complete), the problem is to find a subset A' in a given set A such that $\sum_{a_i \in A'} w(a_i) = \sum_{a_i \in A-A'} w(a_i)$, where each element a_i in A is associated with a size $w(a_i) \in Z^+$. Let U_{max} be an arbitrarily positive real and $U_{max} \neq \infty$. The instance of the PARTITION problem could be reduced to an instance of the MME problem such that a unique task τ_i is created for each element $a_i \in A$, and the required CPU cycles for τ_i is $w(a_i) \cdot U_{max}$. All tasks are ready at time 0, and the common deadline is set as $D = \frac{\sum_{a_i \in A} w(a_i)}{2}$. Let the number M of processors be 2. By applying the approximation algorithm *ALG* to the resulting instance of the MME problem, the energy consumption of the derived schedule would be bounded by the multiplication of $(1+\epsilon)$ and the energy consumption of an optimal schedule if there exists would be any feasible schedule. However, any feasible schedule could not execute tasks at a speed over U_{max}. In other words, if there exists a feasible schedule, then *ALG* must already identify a subset A' of A such that $\sum_{a_i \in A'} w(a_i) = \sum_{a_i \in A-A'} w(a_i)$. If there does not exist a feasible schedule, then *ALG* would report the failure by returning ∞. Since *ALG* is

3 $f()$ is strict convex if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ for any $\alpha \in (0,1)$ and any x, y. For example, $P(s) \propto s^3$ when $s \geq 0$.

a polynomial-time algorithm, such a conclusion contradicts with the NP-Completeness of the PARTITION problem (unless P=NP). ■

Theorem 2 implies that there does not exist any polynomial-time approximation algorithm for the MME problem when $U_{max} \neq \infty$ unless $P = NP$, since ϵ could be any positive real.

3. Multiprocessor Scheduling without Task Migration

In this section, we present approximation algorithms for the MME problem. We first consider the case in which $U_{max} = \infty$ for two and an arbitrary number of processors, respectively. We then show the proposed algorithms can be proved to bound the maximum processor speed by constant factors, when $U_{max} \neq \infty$. Note that, since all tasks are ready at time 0 and share a common deadline, the tasks assigned on a processor can be executed in any order. That is, the execution order for the tasks assigned on a processor does not affect the feasibility and the energy consumption for any feasible schedule. The following formula resulted from the convexity of the energy consumption function is used in this section:

$$P(\frac{c'_1}{D})D + P(\frac{c'_2}{D})D \geq P(\frac{c'_3}{D})D + P(\frac{c'_4}{D})D, \quad (2)$$
$$\text{when } c'_1 + c'_2 = c'_3 + c'_4 \text{ and } 0 \leq c'_1 < c'_3 < c'_4 < c'_2.$$

Based on Formula (2) and Lemma 1, it is clear that the executing of a task τ_i on the processor i from time 0 to D at the speed $\frac{c_i}{D}$ results in an optimal schedule, when $|T| \leq M$. In the following of this section, only non-trivial cases, $|T| > M$, are considered.

3.1. Multiprocessor Scheduling over Two Identical Processors When $U_{max} = \infty$

We shall show how to obtain a fully polynomial time approximation scheme (FPTAS) for two identical processors by a reduction to the MAXIMUM SUBSET SUM problem [4].[4] Given a set A of positive numbers $a_1, a_2, \cdots, a_{|A|}$ and an arbitrary number W, the MAXIMUM SUBSET SUM problem [4] (which is NP-hard) is to find a subset A' of A such that $\sum_{a_i \in A'} a_i \leq W$ and $\sum_{a_i \in A'} a_i$ is maximized.

Lemma 2 (Ibarra and Kim [9]) *The* MAXIMUM SUBSET SUM *problem admits a fully polynomial-time $\frac{1}{(1-\delta)}$-approximation algorithm* SUBSET() *for any $0 < \delta < 1$, where the time complexity is $O(|A|(\frac{3}{\delta})^2)$ and the space complexity is $O(|A| + (\frac{3}{\delta})^3)$.*

4 An algorithm A for a minimization problem is an FPTAS if A is executed in polynomial time in the size of the input and $\frac{1}{\epsilon}$, and the approximation ratio of algorithm A is $1+\epsilon$ [13], where $0 < \epsilon$ is a user input parameter. Note that the approximation ratio is $\frac{1}{1-\epsilon}$ for a maximization problem, where $0 < \epsilon < 1$.

Due to Lemma 1 and non-migration of tasks, there must exist an optimal schedule for the MME problem with two processors which assigns two subsets T^1 and T^2 ($T^1 \cup T^2 = T$) of tasks on the processors 1 and 2 at the speeds $\frac{\sum_{\tau_i \in T^1} c_i}{D}$ and $\frac{\sum_{\tau_i \in T^2} c_i}{D}$, respectively. Without loss of generality, let $\sum_{\tau_i \in T^1} c_i \leq \sum_{\tau_i \in T^2} c_i$. Because of the convexity of the energy consumption functions in Formula (2), achieving the optimal schedule for the MME problem is to generate a subset T^1 of T such that $\sum_{\tau_i \in T^1} c_i \leq \frac{1}{2} \sum_{\tau_i \in T} c_i$ and $\sum_{\tau_i \in T^1} c_i$ is maximized. We develop Algorithm BASIC for the MME problem with two processors by applying the SUBSET routine in Lemma 2 with a proper δ. The input parameter ϵ in Algorithm BASIC is a specified amount of error tolerant to users, which is a necessary requirement for an FPTAS. It is obvious that the correctness of Algorithm BASIC is guaranteed. In the following theorems, we show that setting $\delta = \sqrt{\frac{\epsilon}{2}}$ leads Algorithm BASIC to be a fully polynomial time $(1+\epsilon)$-approximation algorithm for the MME problem when the energy consumption function satisfies Formula (1).

Algorithm 1 : BASIC

Input: (T, D, ϵ);

Output: A feasible schedule SC with minimal energy consumption;

```
1: let $W = \frac{\sum_{\tau_i \in T} c_i}{2}$;
2: $C'$ = SUBSET $(c_1, c_2, \cdots, c_{|T|}, W, \delta)$ with $\delta = \sqrt{\epsilon/2}$; let
   $T^1$ be the corresponding task set of $C'$.
3: output the schedule $SC$ which executes all tasks in $T^1$ at
   the speed $\frac{\sum_{\tau_i \in T^1} c_i}{D}$ on the processor 1 and all tasks in
   $T - T^1$ at the speed $\frac{\sum_{\tau_i \in T - T^1} c_i}{D}$ on the processor 2;
```

Lemma 3 $f(x, \gamma) = \frac{(\gamma x)^3 + (1-\gamma x)^3}{x^3 + (1-x)^3} \leq 2\gamma^2 - 4\gamma + 3$ *for any fixed γ, where $1 \geq \gamma > 0$ and $\frac{1}{2} \geq x > 0$.*

Proof: It is solved when $\frac{\partial f(x,\gamma)}{\partial x} = 0$. ■

Theorem 3 *Algorithm* BASIC *is a $(1+\epsilon)$-approximation algorithm for the* MME *problem for any $0 < \epsilon < 2$ when $P(s) \propto s^3$, i.e., $V_t = 0$ in Formula (1).*

Proof: Let OPT denote a subset of T, where $\sum_{\tau_i \in OPT} c_i \leq W$ and $\sum_{\tau_i \in OPT} c_i$ is maximized. For the simplicity of representation, we use $C(X)$ to denote $\sum_{\tau_i \in X} c_i$ for any subset X of tasks. Let SC_{opt} be the schedule which executes the tasks in OPT at the speed $\frac{C(OPT)}{D}$ on the processor 1 and the tasks in $T - OPT$ at the speed $\frac{C(T-OPT)}{D}$ on the processor 2. SC_{opt} is an optimal solution for the MME problem and

$$\Phi(SC_{opt}) = (P(\frac{C(OPT)}{D}) + P(\frac{C(T) - C(OPT)}{D}))D.$$

Without loss of generality, let $C(OPT) = C(T) \cdot x$ and $C(T - OPT) = C(T) \cdot (1 - x)$. Since $C(OPT) \leq C(T - OPT)$, we have $0 < x \leq \frac{1}{2}$. We know $C(T^1) = \gamma \cdot C(OPT)$, where $1 \geq \gamma \geq 1 - \delta$ due to the approximation ratio of Algorithm SUBSET. The ratio of the energy consumption of SC to that of SC_{opt} is defined as a function $f()$:

$$f(x, \gamma) = \frac{\Phi(SC)}{\Phi(SC_{opt})} = \frac{(\gamma x)^3 + (1 - \gamma x)^3}{x^3 + (1 - x)^3} \leq 2\gamma^2 - 4\gamma + 3,$$

where the inequality comes from Lemma 3. Note that both γ and x are unknown during the calculation. $f(x, 1 - \sqrt{\frac{\epsilon}{2}}) \leq 1 + \epsilon$ by solving $1 + \epsilon = 2\gamma^2 - 4\gamma + 3$. Since $2\gamma^2 - 4\gamma + 3$ is a decreasing function of γ for any $0 < \gamma \leq 1$, $f(x, \gamma) \leq 1 + \epsilon$ if $\delta = \sqrt{\frac{\epsilon}{2}}$.

Therefore, by setting $\delta = \sqrt{\frac{\epsilon}{2}}$, we conclude that Algorithm BASIC is a $(1 + \epsilon)$-approximation algorithm for the MME problem. The time complexity of Algorithm BASIC is $O(|T|\frac{18}{\epsilon})$, and the space complexity is $O(|T| + (\frac{18}{\epsilon})^{1.5})$. ■

We can also prove that Algorithm BASIC is an FPTAS even when $V_t \neq 0$ in Formula (1) in the following theorem.

Theorem 4 *Algorithm* BASIC *is a* $(1+\epsilon)$*-approximation algorithm for the* MME *problem for any* $0 < \epsilon < 2$ *when* $P(s) = C_{ef}(\frac{s^3}{2k^2} + \frac{2V_t s^2}{k} + sV_t^2 + \frac{s^2}{k}\sqrt{\frac{V_t s}{k} + \frac{s^2}{4k^2}} + V_t s\sqrt{\frac{V_t s}{4k} + \frac{s^2}{k^2}})$*, i.e.,* $V_t \neq 0$ *in Formula* (1).

3.2. Multiprocessor Scheduling over an Arbitrary Number of Processors When $U_{max} = \infty$

In this section, we present a scheduling algorithm with a 1.13-approximation ratio for the MME problem when the maximum available processor speed is infinite. Our proposed algorithm shown in Algorithm 2 (Algorithm LTF) adopts the *Largest-Task-First* strategy. That is, tasks are considered in a non-increasing order of their computation requirements.

Let p_m denote the *load* on the processor m. The *load* of a processor is defined as the total amount of the computation requirements of the tasks assigned to that processor. Let T_m denote the set of the tasks assigned to the processor m. Note that the task set T is a sorted set in a non-increasing order of the computation requirement of each task, i.e., $c_i \geq c_j$ if $i < j$. Algorithm LTF assigns a task to the processor with the smallest load by the task order in T. To achieve the minimal energy consumption, based on Lemma 1, each task on the processor m should be executed at the speed $\frac{\sum_{\tau_i \in T_m} c_i}{D}$. The time complexity of Algorithm LTF is $O(|T| \log |T|)$, which is dominated by the sorting of the tasks. Since each task is assigned to one processor without missing the common deadline, the correctness of Algorithm LTF is guaranteed. For the simplicity of representation, the schedule derived from Algorithm LTF is denoted as $SC_{T,LTF}$.

Algorithm 2 : LTF

```
Input: $(T, D, M)$;
Output: A feasible schedule $SC_{T,LTF}$ with minimal energy consumption;
1: sort all tasks in a non-increasing order of the computation requirement of each task;
2: set $p_1, p_2, \cdots, p_M$ to 0, and $T_1, T_2, \cdots, T_M$ to $\phi$;
3: for $i = 1$ to $|T|$ do
4:     find the smallest $p_m$; (break ties arbitrarily)
5:     $T_m \leftarrow T_m \cup \{\tau_i\}$ and $p_m \leftarrow p_m + c_i$;
6: return the schedule $SC_{T,LTF}$ which executes all of the tasks in $T_m$ $(1 \leq m \leq M)$ at the speed $\frac{p_m}{D}$ on the processor $m$;
```

Lemma 4 *Algorithm* LTF *is an optimal algorithm if* $|T| \leq 2M$ *and* $c_{i+M} \geq \frac{1}{2}c_{M-i+1}$ *for all* $1 \leq i \leq |T| - M$.

Proof: It can be proved by transforming any feasible solution into $SC_{T,LTF}$ without increasing the energy consumption. ■

The next step is to derive the lower bound of the MME problem by relaxing the problem constraint. Let k be the largest index satisfying $M \leq k \leq 2M$ and $c_{i+M} \geq \frac{1}{2}c_{M-i+1}$ for all $1 \leq i \leq k - M$. T' represents the set of the first k tasks of T. Note that, if $|T'| < 2M$ and $T - T' \neq \phi$, we know $c_{|T'|+1} < \frac{1}{2}c_{2M-|T'|}$. We relax the constraint of the MME problem so that any task in $T - T'$ could be executed on more than one processor simultaneously. Below, we describe the scheduling method for the relaxed MME problem. We assign the tasks in T' according to Algorithm LTF. Let p_m denote the load of the processor m after performing the task assignment. There exists a positive value P_{min} that satisfies the following equation:

$$\sum_{m=1}^{M}(P_{min} - p_m)\delta_m = \sum_{\tau_i \in T - T'} c_i, \qquad (3)$$

where δ_m is 1 if $P_{min} > p_m$ and 0 otherwise. Since task migration and simultaneous execution of a task on multiple processors are allowed for the tasks in $T - T'$, we can distribute the computation of these tasks among

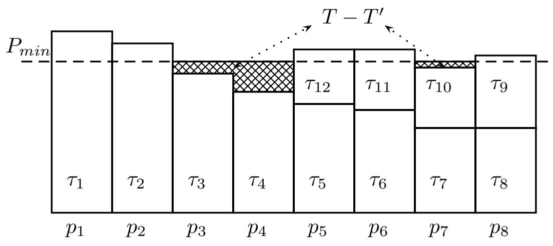

Figure 1. The task assignment of $SC'_{T,LTF}$ for $M = 8$ and $|T'| = 12$. The computation requirements of the tasks in $T - T'$ are distributedx over the processors $3, 4,$ and 7 (the patterned regions).

the processors. If $P_{min} > p_m$, $(P_{min} - p_m)$ CPU cycles of $T - T'$ are distributed on the processor m. Each processor m then performs computation at the speed $\frac{p_m}{D}$ if $p_m > P_{min}$ and $\frac{P_{min}}{D}$ otherwise. Let $SC'_{T,LTF}$ denote the resulting schedule. Figure 1 illustrates the loads of processors in $SC'_{T,LTF}$ for the relaxed MME problem. Due to the optimality provided in Lemma 4, it is clear that $SC'_{T,LTF}$ consumes no more energy than $SC_{T,opt}$, where $SC_{T,opt}$ is an optimal schedule for the MME problem.

Lemma 5 $\frac{\Phi(SC_{T,LTF})}{\Phi(SC_{T,opt})} \leq \frac{\Phi(SC_{T,LTF})}{\Phi(SC'_{T,LTF})} \leq R^*$, *where* $R^* = \text{MAX}\{\frac{\sum_{l_i \in L} P(\frac{l_i}{D})}{M \cdot P(\frac{S}{MD})}\}$ *for any positive integer* M, *positive reals* S *and* D, *and any set* L *of* M *positive reals that satisfy* $\sum_{l_i \in L} l_i = S$ *and* $max_{l_i \in L} l_i \leq \frac{3}{2} min_{l_i \in L} l_i$.

Proof: Since $\Phi(SC'_{T,LTF}) \leq \Phi(SC_{T,opt})$, the first inequality is proved. Let $o_1, o_2, \cdots, o_M$ denote the load on each processor for the task assignment generated by Algorithm LTF for T' and $p_1, p_2, \cdots, p_M$ for T. max_p, min_p, and min_o are the values with $max\{p_i\}$, $min\{p_i\}$, and $min\{o_i\}$, respectively. It is clear that $max_p \geq P_{min}$, $min_p \leq P_{min}$, and $\Phi(SC'_{T,LTF}) \geq MD \cdot P(\frac{\sum_{\tau_i \in T} c_i}{MD})$. If $max_p = P_{min}$, then $min_p = P_{min}$ and Algorithm LTF generates an optimal solution. Therefore, we only consider the condition where $max_p > P_{min} \geq min_p$ and $T - T' \neq \phi$. We now prove the second inequality. We first consider the case where $o_m \leq P_{min}$ for each processor m. Let m^* be the processor with the largest load in $SC_{T,LTF}$, i.e., $p_{m^*} = max_p$, and τ_k be the last task added into T_{m^*} in $SC_{T,LTF}$. Once a processor m satisfies $p_m \geq P_{min}$, Algorithm LTF will not assign any more task to the processor m. Therefore, we have $\sum_{\tau_i \in T_{m^*} - \{\tau_k\}} c_i < P_{min}$. It is clear that $min_p \geq \sum_{\tau_i \in T_{m^*} - \{\tau_k\}} c_i$; otherwise, Algorithm LTF will not assign task τ_k to the processor m^*. Therefore, $max_p - min_p \leq c_k$. Because of $o_m \leq P_{min}$ for each processor m, we have $c_k \leq c_{|T'|+1}$. We know $min_o \leq min_p$ since Algorithm LTF adds the tasks in $T - T'$ to the processor with the minimal load. Due to the definition of T', $c_{|T'|+1} \leq \frac{1}{2} min_o$.[5] Combining all inequality relations mentioned above, we have

$$max_p - min_p \leq c_k \leq c_{|T'|+1} \leq \frac{1}{2} min_o \leq \frac{1}{2} min_p.$$

Therefore, $max_p \leq \frac{3}{2} min_p$ which proves the second inequality.

We now consider the case where some $o_m > P_{min}$. Let these processors be J, where $|J| \geq 1$. For each processor m in J, $SC_{T,LTF}$ and $SC'_{T,LTF}$ assign the same tasks on the processor m, e.g., the processors $1, 2, 5, 6,$ and 8 in Figure 1. By assuming $\frac{\Phi(SC_{T,LTF})}{\Phi(SC'_{T,LTF})} > R^*$, we conclude that $\frac{\Phi(SC_{T,LTF}) - \sum_{j \in J} P(o_j/D)D}{\Phi(SC'_{T,LTF}) - \sum_{j \in J} P(o_j/D)D} > R^*$. This contradicts the case for $M = M - |J|$ and $T = T - \cup_{j \in J} T_j$. Therefore, the approximation ratio for Algorithm LTF is R^*. ■

We now derive the value of R^* when the energy consumption function satisfies Formula (1).

Theorem 5 *Algorithm* LTF *is a 1.13-approximation algorithm for the* MME *problem when* $P(s) \propto s^3$, *i.e.,* $V_t = 0$ *in Formula* (1).

Proof: We prove this theorem by showing that $R^* \leq 1.13$. By the definition of R^* in Lemma 5, there must exist at least one real number x for a set L, where $2x \leq l_i \leq 3x$ for all $l_i \in L$. It is clear that $2xM \leq S \leq 3xM$. For each element $l_i \in L$, we know $P(\frac{l_i}{D}) \leq \frac{3x - l_i}{x} P(\frac{2x}{D}) + \frac{l_i - 2x}{x} P(\frac{3x}{D})$.[6] Therefore, we have $\sum_{l_i \in L} P(\frac{l_i}{D}) \leq kP(\frac{3x}{D}) + (M-k)P(\frac{2x}{D})$ for a real k satisfying $k = \frac{S - 2Mx}{x}$ (rephrasing of $3x \cdot k + 2x \cdot (M-k) = S$). R^* is obtained by finding the proper x which maximizes the function $f(x) = kP(\frac{3x}{D}) + (M-k)P(\frac{2x}{D})$. Without loss of generality, let $P(s) = \alpha s^3$, where α is a constant. By solving $f'(x) = 0$ and showing $f''(x) < 0$ for all x satisfying $\frac{S}{2M} \geq x \geq \frac{S}{3M}$, $f(x)$ is maximized when $x = \frac{19S}{45M}$ and the maximum value of $f(x)$ is $\alpha \frac{19^3 S^3 / M^2}{3 \cdot 45^2 D^3} \approx 1.13 \alpha \frac{S^3}{M^2 D^3}$. Therefore, $R^* \leq 1.13$. ■

Corollary 1 *Algorithm* LTF *is a 1.13-approximation algorithm for the* MME *problem when* $P(s) = C_{ef}(\frac{s^3}{2k^2} + \frac{2V_t s^2}{k} + sV_t^2 + \frac{s^2}{k}\sqrt{\frac{V_t s}{k} + \frac{s^2}{4k^2}} + V_t s \sqrt{\frac{V_t s}{4k} + \frac{s^2}{k^2}})$.

5 There are two cases: 1. If $min_o = c_{2M-|T'|}$, then $c_{|T'|+1} < \frac{1}{2} c_{2M-|T'|} = \frac{1}{2} min_o$; 2. If $min_o = c_i + c_j$ for some $i, j > 2M - |T'|$, then relations $c_{|T'|+1} \leq c_i$ and $c_{|T'|+1} \leq c_j$ result in $c_{|T'|+1} \leq \frac{1}{2} min_o$.

6 The inequality comes from that $P(\alpha x + (1-\alpha)y) \leq \alpha P(x) + (1-\alpha)P(y)$ for any $\alpha \in (0,1)$ and any x, y. The coefficients of $P(\frac{2x}{D})$ and $P(\frac{3x}{D})$ are obtained by solving a, b in the following equations: $l_i = a \cdot 2x + b \cdot 3x$ and $1 = a + b$.

3.3. Multiprocessor Scheduling When $U_{max} \neq \infty$

By adopting the constraint-violation approach [11], we propose an artificial-bound approach by first setting an artificial upper bound on the processor speed and then derive feasible schedules in the minimization of energy consumption. We show that Algorithms BASIC and LTF bound the maximum processor speed by the factors of $\sqrt{\frac{\epsilon}{2}}$ and $(\frac{4}{3} - \frac{1}{3M})$, respectively. For the simplicity of representation, we assume that tasks in T are sorted in a non-increasing order of their computation requirements. We first prove that Algorithm LTF could derive a schedule with a 1.13-approximation ratio without violating the maximum processor speed for certain input instances (Please see Theorem 6):

Theorem 6 *Algorithm* LTF *is a 1.13-approximation algorithm if the given input instance satisfies* $\frac{\sum_{\tau_i \in T} c_i}{M} + c_{M+1} \leq U_{max}D$ *and* $c_1 \leq U_{max}D$.

Proof: This theorem is proved by contradiction. We assume that there exists a processor m in $SC_{T,LTF}$, where $\sum_{\tau_i \in T_m} c_i > U_{max}D$. Let τ_k be the last task added into T_m in Algorithm LTF. Two cases are considered. If $k \leq M$, we know $c_1 \geq c_k > U_{max}D$. This contradicts the assumption. If $k > M$, we have $\sum_{\tau_i \in T_m - \{\tau_k\}} c_i > U_{max}D - c_k \geq U_{max}D - c_{M+1}$. In Algorithm LTF, once $p_m \geq \frac{\sum_{\tau_i \in T} c_i}{M}$, no more tasks can be assigned on the processor m. Therefore, $\frac{\sum_{\tau_i \in T} c_i}{M} > \sum_{\tau_i \in T_m - \{\tau_k\}} c_i$. Based on the above inequalities, we have $\frac{\sum_{\tau_i \in T} c_i}{M} + c_{M+1} > U_{max}D$. This contradicts our assumption. ■

We now show that Algorithms BASIC and LTF bound the maximum processor speed by the factors of $\sqrt{\frac{\epsilon}{2}}$ and $(\frac{4}{3} - \frac{1}{3M})$, respectively.

Theorem 7 *Given an input instance with a feasible schedule for the* MME *problem, no schedule derived from Algorithm* LTF *uses any processor speed larger than* $(\frac{4}{3} - \frac{1}{3M})U_{max}$.

Proof: Let O be a feasible schedule for the input instance. Without loss of generality, O partitions T into M disjoint subsets of tasks. We assume that m is the processor with the largest load in O. Since O is a feasible schedule, the load on the processor m, says p_m, must be no more than $U_{max}D$. Let n be the processor with the largest load in $SC_{T,LTF}$. By rephrasing the processing time into computation requirement in the Makespan problem, we know that $p_n \leq (\frac{4}{3} - \frac{1}{3M})p_m \leq (\frac{4}{3} - \frac{1}{3M})U_{max}D$ since the *Longest-Processing-Time-First* algorithm was proved to be a $(\frac{4}{3} - \frac{1}{3M})$-approximation algorithm for the Makespan problem in [6].[7] We complete the proof. ■

Theorem 8 *Given an input instance with a feasible schedule for the* MME *problem over two processors, no schedule derived from Algorithm* BASIC *uses any processor speed larger than* $(1 + \sqrt{\frac{\epsilon}{2}})U_{max}$.

Proof: It comes from the setting of δ in Algorithm BASIC. ■

4. Task Migration: An Optimal Algorithm

Algorithm 3 : LTF-M

```
Input: $(T, D, M)$;
Output: An optimal schedule with minimum energy consumption;
 1: sort $T$ in a non-increasing order of the computation requirement of each task; let $C \leftarrow \sum_{\tau_i \in T} c_i$;
 2: if $\frac{C}{MD} > U_{max}$ or $\exists \tau_i \in T$ such that $\frac{c_i}{D} > U_{max}$ then
 3:     return non-existence of any feasible schedule;
 4: let $i \leftarrow 1$;
 5: while $i \leq |T|$ do
 6:     if $c_i > \frac{C}{M}$ then
 7:         schedule $\tau_i$ to be executed at the speed $\frac{c_i}{D}$ on the processor $M$ from time 0 to $D$;
 8:         $C \leftarrow C - c_i$, $i \leftarrow i + 1$, and $M \leftarrow M - 1$;
 9:     else
10:         break;
11: let $S \leftarrow \frac{C}{MD}$ and $t \leftarrow 0$;
12: while $i \leq |T|$ do
13:     if $t + \frac{c_i}{S} > D$ then
14:         schedule $\tau_i$ to be executed at the speed $S$ on the processor $M - 1$ from time 0 to $t + \frac{c_i}{S} - D$ and on the the processor $M$ from time $t$ to $D$; $M \leftarrow M - 1$;
15:     else
16:         schedule $\tau_i$ to be executed on the processor $M$ at the speed $S$ from time $t$ to $t + \frac{c_i}{S}$;
17:     $i \leftarrow i + 1$ and $t \leftarrow (t + \frac{c_i}{S}) \mod D$;
18: return the schedule of all tasks;
```

In this section, an efficient optimal algorithm is proposed for the MMEM problem, where task migration is allowed. If $|T| \leq M$, based on Formula (2) shown in Section 3, it is clear that the executing of each task τ_i on the processor i from time 0 to D at the speed $\frac{c_i}{D}$ results in an optimal schedule. Our proposed Algorithm LTF-M (Algorithm 3) adopts the Largest-Task-First strategy again, and the time complexity $O(|T| \log |T|)$ comes from the sorting of T in line 1. We can prove the following two lemmas.

7 The Makespan problem is as follows: Given processing time for n tasks, find an assignment of the tasks to M identical processors so that the completion time for these tasks is minimized.

Lemma 6 *If $c_1 > SD$ and $|T| \geq M$, then there exists an optimal schedule which executes only τ_1 on a processor at the speed $\frac{c_1}{D}$ from 0 to D, where $S = \frac{\sum_{\tau_i \in T} c_i}{MD}$.*

Lemma 7 *If $c_1 \leq SD$ and $|T| \geq M$, then there exists an optimal schedule which executes each task in T on at most two processors at the speed S, where $S = \frac{\sum_{\tau_i \in T} c_i}{MD}$.*

Theorem 9 *Any schedule derived from Algorithm* LTF-M *is an optimal schedule.*

Proof: If $c_1 > SD$ where $S = \frac{\sum_{\tau_i \in T} c_i}{MD}$, then Algorithm LTF-M executes τ_1 on M at the speed $\frac{c_1}{D}$, and the remaining tasks $T - \{\tau_1\}$ and $M-1$ processors form a subproblem of the MMEM problem; otherwise, Algorithm LTF-M executes each task in T over at most two processors at the speed S. Based on Lemmas 6 and 7, we conclude this proof by repeating the above procedure in solving the MMEM subproblems. ■

5. Conclusion

This paper targets energy-efficient scheduling of tasks over multiple processors, where tasks share a common deadline. Distinct from the past work, this paper proposes approximation algorithms with different approximation bounds for processors with/without constraints on the maximum processor speed. We show the non-existence of polynomial-time approximation algorithms in the minimization of energy consumption for multiprocessor scheduling over processors with an upper bound on the processor speed, unless $P = NP$. When there is no constraint on processor speeds, we propose an approximation algorithm for two-processor scheduling to provide trade-offs among the specified error, the running time, the approximation ratio, and the memory space complexity. An approximation algorithm with a 1.13-approximation ratio for M-processor systems is also derived ($M > 2$). When there is an upper bound on processor speeds, an artificial-bound approach is taken to minimize the energy consumption with a 1.13-approximation ratio. Furthermore, an optimal polynomial-time scheduling algorithm is proposed for the minimization of the energy consumption when task migration is allowed.

For future research, we shall explore multiprocessor energy-efficient scheduling for task sets with arbitrary deadlines and arrival times.

References

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

[2] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, pages 103–116. ACM Press, 2001.

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

[4] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W.H. Freeman and Co, 1979.

[5] G. Golub and J. Ortega. *Scientific Computing and Differential Equations.* Academic Press, 1992.

[6] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:263–269, 1969.

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

[8] 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.

[9] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *Journal of the ACM*, 22(4):463–468, 1975.

[10] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the International Symposium on Low Power Electroncs and Design*, pages 197–202, 1998.

[11] 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.

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

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

[14] 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.

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

[16] 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.

[17] 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.