Multiprocessor Synthesis for Periodic Hard Real-Time Tasks under a Given Energy Constraint*

Heng-Ruey Hsu, Jian-Jia Chen, and Tei-Wei Kuo
Department of Computer Science and Information Engineering
Graduate Institute of Networking and Multimedia
National Taiwan University, Taipei, Taiwan 106, ROC.
Emails:{b89108, r90079, ktw}@csie.ntu.edu.tw

ABSTRACT

The energy-aware design for electronic systems has been an important issue in hardware and/or software implementations, especially for embedded systems. This paper targets a synthesis problem for heterogeneous multiprocessor systems to schedule a set of periodic real-time tasks under a given energy consumption constraint. Each task is required to execute on a processor without migration, where tasks might have different execution times on different processor types. Our objective is to minimize the processor cost of the entire system under the given timing and energy consumption constraints. The problem is first shown being $\mathcal{NP}$-hard and having no polynomial-time algorithm with a constant approximation ratio unless $\mathcal{NP} = \mathcal{P}$. We propose polynomial-time approximation algorithms with $(m + 2)$-approximation ratios for this challenging problem, where m is the number of the available processor types. Experimental results show that the proposed algorithms could always derive solutions with system costs close to those of optimal solutions.

Keywords: Energy-aware systems, Task scheduling, Real-time systems, Task partitioning, Multiprocessor synthesis.

1. INTRODUCTION

Energy-efficiency has been an important design issue for hardware and/or software implementations, especially for embedded or mobile systems. Energy-efficient designs and scheduling could not only extend the power-on duration of battery-driven devices but also help in cutting down the power bill of server systems significantly [5]. In the past decade, energy-efficient task scheduling with various deadline constraints has received a lot of attention. Various work has been done for uniprocessor task scheduling, such as those in [1, 3, 17]. Different heuristics were also proposed for energy consumption minimization under different task models in multiprocessor environments, e.g., [2, 6, 11, 18]. System-level synthesis for heterogeneous multiprocessor was also explored for cost consideration of consumer markets, e.g., [15, 4]. In particular, the synthesis problem for energy-efficient task scheduling of periodic hard real-time tasks under a given processor cost constraint was first explored in [11], where processors might have different costs. Different variations of synthesis problem for energy-efficient were mentioned in [10].

This work is motivated by the needs in the minimization of the entire system cost under the computing demands of applications and a given energy consumption constraint. Given different implementations (over different hardware platforms/processors), tasks might have different execution times and consume different amounts of energy. Note that processors under discussions could be ASIC chips because of the implementations of algorithms in hardware. For example, the decoding of MPEG files could be done by different digital processing chips with different energy consumption profiles and execution times. For these tasks, we are interested in the determination of the amounts of these two DSP chips of so that the cost of the allocated DSP chips are minimized, and the energy constraint of the system and timing constraints of the tasks are satisfied.

This paper targets energy-efficient scheduling of periodic hard real-time tasks, where no task migration among processors is allowed. The goal is to assign a selected processor to the execution of each task such that the total cost of processors is minimized, and the energy and task timing constraints are satisfied. The problem is $\mathcal{NP}$-hard even when the number of the available processor types is a constant. We show that there does not exist any polynomial-time algorithm with a constant approximation ratio by an L-reduction [14] from the set cover problem, unless $\mathcal{NP} = \mathcal{P}$. An approximation algorithm based on a rounding technique is proposed by applying a parametric relaxation on an integer linear programming (ILP) formula. The approximation ratio is $(m+2)$, where m is the number of the available processor types. The algorithm is then improved with better solutions in many cases but with the same approximation ratio. The performance of the proposed algorithms was evaluated by a series of experiments, compared to optimal solutions derived by exhaustive search algorithms (when the size of the task set is small) or a lower bound derived by a relaxation of the ILP formula of the synthesis problem. Experimental results show that the proposed algorithms could always derive solutions with system costs close to those of optimal solutions.

The rest of this paper is organized as follows: Section 2 formally defines the multiprocessor synthesis problem explored in this paper. Section 3 presents the proposed approximation algorithms. The experimental results for the performance evaluation of the proposed algorithm are presented in Section 4. Section 5 is the conclusion.

2. PROBLEM DEFINITION

This paper is interested in a synthesis problem for heterogeneous multiprocessor environments. We consider an environment with m different types of available processors. Let $\mathcal{M}$ be the set of available processor types. For each type M_i of available processor types in $\mathcal{M}$, an allocation cost C_i is associated with the processor type, where C_i could be the corresponding price, area, or any property under considerations.

We consider the scheduling problem of a set $\mathcal{T}$ of n periodic real-time tasks without dependency constraints [13]. 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 in $\mathcal{T}$ is characterized by four parameters: its *execution time*, *energy consumption*, *period*, and *relative deadline*. $c_{i,j}$ and $e_{i,j}$ denote the worst-case execution time and the energy consumption required to complete any job execution of task τ_i on one processor of M_j, respectively, when

*Support in parts by research grants from ROC National Science Council NSC-94-2752-E-002-008-PAE.

τ_i executes alone and has all the resources that it requires. If there does not exist any implementations of task τ_i on processor M_j, we could assume that both $e_{i,j}$ and $c_{i,j}$ are infinity. The period p_i of task τ_i is the minimal arrival interval between two consecutive jobs of the task. The relative deadline of task τ_i is the longest span of the time interval between the latest completion time and the release time of a job of τ_i. In this paper, the relative deadline of a task is assumed being equal to the period of the task. If $c_{i,j} > p_i$ for some M_j, it is clear that the executing of τ_i on one processor of M_j will unavoidably let τ_i miss its deadline. For such a case, we just set both $e_{i,j}$ and $c_{i,j}$ as infinity. A task completes in time means that all of the jobs of the task completes before their corresponding deadlines. Throughout this paper, we focus our studies on the case in which all of the tasks arrive at time 0. The *hyper-period* of a task set $\mathcal{T}$, denoted by L, is the least common multiple (LCM) of the periods of the tasks in $\mathcal{T}$. For the brevity, let $E_{i,j}$ denote the total energy consumption of task τ_i executing on one processor of the processor type M_j in the hyper-period (Note that there is no task migration). That is, $E_{i,j} = \frac{L}{p_i} e_{i,j}$.

In this paper, we are interested in the joint considerations of the scheduling of hard real-time tasks and the allocation of processors such that all of the tasks in $\mathcal{T}$ complete in time, the total energy consumption of tasks in the hyper-period is no more than a given energy budget E_{budget}, and the cost of the allocated processors is minimized. However, it might be difficult to define the energy constraint in a hyper-period. As a result, we assume that an average value on the power consumption constraint is given. When the average power consumption constraint P_{avg} is given, E_{budget} is set as $P_{avg}L$. The problem considered in this paper is defined as follows:

DEFINITION 1. *Multiprocessor Allocation for Real-Time Tasks under an Energy Constraint (*MARTEC*) problem*

Consider a set $\mathcal{T}$ of independent tasks over a set $\mathcal{M}$ of available processors with m different types, where all of the tasks in $\mathcal{T}$ arrive at time 0, and m is a positive integer. Each task $\tau_i \in \mathcal{T}$ is characterized by its period p_i, where the relative deadline of τ_i is equal to p_i. When τ_i is executed on one processor of processor type $M_j \in \mathcal{M}$, τ_i is associated with its execution time $c_{i,j}$ and its energy consumption $e_{i,j}$ for each job execution on the processor, where $E_{i,j} = \frac{L}{p_i} e_{i,j}$ is defined as its total energy consumption in the hyper-period L. Moreover, we are given an energy budget E_{budget} for the maximum energy consumption in the hyper-period L of $\mathcal{T}$. The objective of the problem is to derive a schedule of $\mathcal{T}$ and a multisubset of $\mathcal{M}$ for processor allocation such that each task in $\mathcal{T}$ is executed on an allocated processor and completes in time, the total energy consumption in the hyper-period of $\mathcal{T}$ does not exceed E_{budget}, and the total cost of allocated processors is minimized. □

Without loss of generality, we assume that $\mathcal{M}$ and $\mathcal{T}$ do not consist of any dominating set, in which the domination of M_i over M_j means that M_i is more expensive than M_j per unit, and any task has a longer execution time and more energy consumption on one processor of M_i than those on another processor of M_j. In such a case, the type of processor M_i is not required.

THEOREM 1. *The* MARTEC *problem is $\mathcal{NP}$-hard in the strong sense.*

PROOF. We could show that the MARTEC problem is $\mathcal{NP}$-hard in the strong sense by a reduction from the bin packing problem, which is $\mathcal{NP}$-complete in the strong sense [8]. □

Due to the $\mathcal{NP}$-hardness of the MARTEC problem, we focus the study on approximation algorithms with a worst-case guarantee on processor allocation cost. Based on [16], a polynomial-time α-approximation algorithm for the MARTEC problem must have a polynomial time complexity of the input size and could derive a solution with the total cost at most α times of an optimal solution, for any input instance, in which α is also referred to as the approximation ratio of the approximation algorithm. The following theorem reveals the inapproximability result of the MARTEC problem when $\mathcal{NP} \neq \mathcal{P}$.

THEOREM 2. *Unless $\mathcal{NP} = \mathcal{P}$, there does not exist any polynomial-time approximation algorithm with a constant approximation ratio.*

PROOF. This theorem is proved by an L-reduction [14, §13] from the set cover problem, which does not admit any polynomial-time approximation algorithm with a constant approximation ratio unless $\mathcal{NP} = \mathcal{P}$ [16, §29]. We omit the detail due to the space limitation. □

3. APPROXIMATION ALGORITHMS

In this section, the MARTEC problem is first formulated as an integer linear programming, and a series of relaxations is then performed to derive a feasible schedule and a proper allocation of processors in polynomial time. We will show that the proposed algorithms could derive approximated solutions with the worst-case guarantee. We would only focus the discussions on input instances in which $\sum_{\tau_i \in \mathcal{T}} \min_{M_j \in \mathcal{M}} e_{i,j} \leq E_{budget}$, since there does not exist any feasible solution for the other cases.

3.1 Integer Programming and Relaxations

As shown in [12], the earliest-deadline-first (EDF) scheduling algorithm is an optimal uniprocessor scheduling algorithm for independent real-time tasks. A task set is schedulable if and only if the total utilization of the task set is no more than 100%, where the *utilization* of a task is defined as its execution time divided by its period. In this paper, we consider the joint scheduling and allocation problem, in which EDF scheduling is applied to each allocated processor. Suppose that the number of allocated processors of processor type M_j is K_j. For each task τ_i in $\mathcal{T}$, a binary variable $z_{i,j,k}$ is set as 1 if τ_i is assigned to execute on the k-th allocated processor of processor type M_j; otherwise, $z_{i,j,k} = 0$. The set $\mathcal{T}_{j,k}$ of tasks assigned onto the k-th allocated processor of M_j is schedulable by EDF if $\sum_{\tau_i \in \mathcal{T}_{j,k}} u_{i,j} \leq 1$, where $u_{i,j}$ is the utilization of τ_i on a processor of M_j, i.e., $\frac{c_{i,j}}{p_i}$. The MARTEC problem is formulated as an integer linear programming problem as follows:

$$
\begin{array}{ll}
\text{minimize} & \sum_{M_j \in \mathcal{M}} K_j \cdot C_j \\
\text{subject to} & \sum_{M_j \in \mathcal{M}} \sum_{\tau_i \in \mathcal{T}} \sum_{k=1}^{n} E_{i,j} \cdot z_{i,j,k} \leq E_{budget}, \\
& \sum_{M_j \in \mathcal{M}} \sum_{k=1}^{K_j} z_{i,j,k} = 1 \ \ , \forall \tau_i \in \mathcal{T}, \\
& \sum_{M_j \in \mathcal{M}} \sum_{k=K_j+1}^{n} z_{i,j,k} = 0 \ \ , \forall \tau_i \in \mathcal{T}, \\
& \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot z_{i,j,k} \leq 1 \ \ , \forall M_j \in \mathcal{M}, k = 1 \ldots K_j, \text{ and} \\
& z_{i,j,k} \in \{0,1\}, \forall \tau_i \in \mathcal{T}, \ \forall M_j \in \mathcal{M}, k = 1 \ldots K_j,
\end{array}
\tag{1}
$$

where the first constraint requires that the total energy consumption of all of the tasks is no more than the given energy budget E_{budget}, the second and third constraints require that each task τ_i must execute on one allocated processor only, and the fourth constraint means that the total utilization of the tasks executing on one allocated processor must be no more than one (because of EDF scheduling). However, the derivation of an optimal solution of Equation (1) is a $\mathcal{NP}$-hard problem since the problem could be formulated as a standard integer linear programming problem, which is $\mathcal{NP}$-hard in the strong sense.

Instead of looking for an optimal solution for Equation (1), we could derive an approximated solution in polynomial time by performing a series of relaxations. The first relaxation will be on the objective function to reduce the number of variables required in the programming. For each task τ_i in $\mathcal{T}$, a binary variable $y_{i,j}$ is set as 1 if τ_i is assigned to execute on a processor of processor type

M_j; otherwise, $y_{i,j} = 0$. $\lceil \sum_{i=1}^{n} u_{i,j} \cdot y_{i,j} \rceil$ is an under-estimated number of the required number of processors of processor type M_j. Equation (1) could be relaxed into the following integer linear programming problem:

$$\begin{array}{ll} \text{minimize} & \sum_{M_j \in \mathcal{M}} \left\lceil \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y_{i,j} \right\rceil C_j \\ \text{subject to} & \sum_{M_j \in \mathcal{M}} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y_{i,j} \leq E_{budget}, \\ & \sum_{M_j \in \mathcal{M}} y_{i,j} = 1 \quad, \forall \tau_i \in \mathcal{T}, \text{ and} \\ & y_{i,j} \in \{0,1\}, \forall \tau_i \in \mathcal{T}, \forall M_j \in \mathcal{M}. \end{array} \tag{2}$$

For any feasible solution of Equation (2), each task is assigned to exactly one processor type. Let task set $\mathcal{T}_j$ be the set of the tasks in $\mathcal{T}$ assigned on the processors of processor type M_j for a solution of Equation (2), i.e., $\mathcal{T}_j = \{\tau_i \in \mathcal{T} \mid y_{i,j} = 1\}$. We propose to adopt the first-fit strategy to assign tasks in $\mathcal{T}_j$ to processors of M_j (referred to as Algorithm FF). In each iteration, we assign an un-assigned task τ_i in $\mathcal{T}_j$ to an allocated processor if the resulting total utilization of the tasks assigned on the processor is no more than 100%. If no such an allocated processor exists, we must get a new processor of M_j and assign τ_i to the newly allocated processor. The time complexity of Algorithm FF is $O(|\mathcal{T}_j|^2)$. Algorithm FF was shown being a 2-approximation algorithm of the bin packing problem [16, §9]. The following lemma shows that the number of the allocated processors of a solution derived by Algorithm FF is at most $\max\{1, 2\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}\}$ for any $\mathcal{T}_j$.

LEMMA 1. *Given a task set $\mathcal{T}_j$, the number of the allocated processors of processor type M_j in Algorithm* FF *is at most* $\max\{1, 2\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}\}$.

PROOF. There are two cases to consider: If $\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}$ is no more than 1, Algorithm FF would allocate only one processor and assign all of the tasks in $\mathcal{T}_j$ on that processor. Therefore, only one processor of M_j is allocated in this case. The other case is when $\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}$ is greater than 1: Suppose that the number of the allocated processors of processor type M_j by Algorithm FF is κ. κ must be at least 2 because of the total utilization on the processor type. We shall show that $\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}$ is no less than $\frac{1}{2}\kappa$. In Algorithm FF, there must be at most one allocated processor with utilization less than $1/2$; otherwise, the workloads on two such processors should be left on one. Let u^* be the total utilization of the allocated processor with utilization less than $1/2$. All of the other $\kappa - 1$ allocated processors must have total utilization no less than $1 - u^*$. Otherwise, the solution contradicts Algorithm FF. We know that $(\sum_{\tau_i \in \mathcal{T}_j} u_{i,j} \geq u^* + (\kappa - 1)(1 - u^*) \geq \kappa/2)$ since $u^* < 1/2$ and $\kappa \geq 2$. As a result, $\kappa \leq 2\sum_{\tau_i \in \mathcal{T}_j} u_{i,j}$. □

According to Lemma 1, the number of the allocated processors of M_j under Algorithm FF is at most twice of the ceiling of the total utilization of the tasks assigned on processors of M_j. Suppose that the vector of $y'_{i,j}$ is an optimal solution for Equation (2). The assignment of tasks in $\mathcal{T}_j = \{\tau_i \in \mathcal{T} \mid y'_{i,j} = 1\}$ to the allocated processors of M_j under Algorithm FF for all M_j in $\mathcal{M}$ will not violate the energy constraint E_{budget}, and EDF schedules all of the tasks in $\mathcal{T}$ in time. Since $(\sum_{M_j \in \mathcal{M}} \left\lceil \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y'_{i,j} \right\rceil C_j)$ is a lower bound of an optimal solution of the MARTEC problem, we could have a 2-approximation algorithm for the MARTEC problem by applying Lemma 1. However, the deriving of an optimal solution for Equation (2) is also $\mathcal{NP}$-hard, since Equation (2) could be considered as an integer linear programming problem. When we relax the integral constraint of $y_{i,j}$ so that $y_{i,j}$ could be any fractional number, a naive relaxation of Equation (2) is as follows:

	p_i	$c_{i,j}$		$e_{i,j}$	
		M_1	M_2	M_1	M_2
τ_1	$50ms$	$30ms$	$50ms$	EmJ	$1mJ$
τ_2	$100ms$	$60ms$	$100ms$	$2EmJ$	$2mJ$

$C_1 = 1, C_2 = B$, and $E_{budget} = 4E - 1mJ$

Table 1: An input instance for demonstrating the unbounded relaxation of Equation (3)

$$\begin{array}{ll} \text{minimize} & \sum_{M_j \in \mathcal{M}} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y_{i,j} \cdot C_j \\ \text{subject to} & \sum_{M_j \in \mathcal{M}} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y_{i,j} \leq E_{budget}, \\ & \sum_{M_j \in \mathcal{M}} y_{i,j} = 1 \quad, \forall i = 1 \ldots n, \text{ and} \\ & y_{i,j} \geq 0, \forall \tau_i \in \mathcal{T}, \forall M_j \in \mathcal{M}. \end{array} \tag{3}$$

An optimal solution of Equation (3) provides a new lower bound of an optimal solution of the MARTEC problem. We shall show that such a lower bound might be far away from an optimal solution of the MARTEC problem in the worst case:

LEMMA 2. *There exists an input instance such that the gap between the cost of an optimal solution of the input instance and an optimal solution of the relaxation in Equation (3) is unbounded.*

PROOF. This lemma can be proved by providing an example. Suppose that we are given two tasks τ_1 and τ_2 in $\mathcal{T}$ with parameters listed in Table 1. The cost of an optimal solution to Equation (3) is $(1 + \frac{2E-3}{2E-2})0.6 + \frac{B}{2E-2}$ by setting $(y_{1,1}, y_{1,2}, y_{2,1}, y_{2,2})$ as $(1, 0, \frac{2E-3}{2E-2}, \frac{1}{2E-2})$. Obviously, if both tasks τ_1 and τ_2 are assigned to execute on processors of M_1, the total energy consumption is $4EmJ$, which is more than E_{budget} $4E - 1mJ$. Hence, an optimal assignment allocates one processor of M_1 and one processor of M_2 with a total price equal to $1 + B$. When the energy consumption E is large enough, the gap between the cost of an optimal solution for the input instance and an optimal solution of the relaxation in Equation (3) is $\frac{1+B}{1.2}$. Hence, the gap is unbounded. □

3.2 Approximation Algorithms Based on Parametric Rounding

This subsection shows that we could relax Equation (2) in a parametric way so that the gap between an optimal solution of Equation (2) and an optimal solution of the relaxed problem is bounded for any input instance. Moreover, the proposed algorithm will derive a feasible solution for the MARTEC problem by referring to an optimal solution of the relaxed problem. For any input instance, we show that the proposed algorithm will derive a solution with at most $(m + 2)$ times allocation cost of an optimal solution of the relaxed problem, which is a lower bound of feasible solutions of the MARTEC problem.

First, we re-index the available processor types in $\mathcal{M}$ so that $C_1 \leq C_2 \leq \cdots \leq C_m$. The idea behind the parametric relaxation of Equation (2) is that we restrict the solution of the input instance. When the parameter is specified as m', the solution of the input instance is not to use any processor of M_j with $j > m'$ and to use at least one processor of $M_{m'}$. Clearly, the minimum solution of the following integer programming problem among $m' = 1, 2, \ldots, m$ is a lower bound of the MARTEC problem:

$$\begin{array}{ll} \text{minimize} & \left\lceil \sum_{\tau_i \in \mathcal{T}} u_{i,m'} \cdot y_{i,m'} \cdot C_{m'} \right\rceil \\ & + \sum_{j=1}^{m'-1} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y_{i,j} \cdot C_j \\ \text{subject to} & \sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} > 0, \\ & \sum_{j=1}^{m'} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y_{i,j} \leq E_{budget}, \\ & \sum_{j=1}^{m'} y_{i,j} = 1 \quad, \forall \tau_i \in \mathcal{T}, \text{ and} \\ & y_{i,j} \in \{0,1\}, \forall \tau_i \in \mathcal{T}, \forall j = 1, 2, \cdots m'. \end{array} \tag{4}$$

For each specified m', we relax Equation (4) by relaxing the integral constraint of $y_{i,j}$ so that $y_{i,j}$ could be any fractional number. The ceiling of $\sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'}$ and the constraint $\sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} > 0$ could be relaxed by two cases when $\sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} \geq 1$ or $\sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} \leq 1$. As a result, for each m', we could relax Equation (4) into the following two subequations:

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{m'} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y_{i,j} \cdot C_j \\ \text{subject to} & \sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} \geq 1, \\ & \sum_{j=1}^{m'} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y_{i,j} \leq E_{budget}, \\ & \sum_{j=1}^{m'} y_{i,j} = 1 \ \ , \forall \tau_i \in \mathcal{T}, \text{ and} \\ & y_{i,j} \geq 0, \forall \tau_i \in \mathcal{T}, \forall j = 1, 2, \cdots m'; \end{array} \tag{5a}$$

$$\begin{array}{ll} \text{minimize} & C_{m'} + \sum_{j=1}^{m'-1} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y_{i,j} \cdot C_j \\ \text{subject to} & \sum_{\tau_i \in \mathcal{T}} y_{i,m'} \cdot u_{i,m'} \leq 1, \\ & \sum_{j=1}^{m'} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y_{i,j} \leq E_{budget}, \\ & \sum_{j=1}^{m'} y_{i,j} = 1 \ \ , \forall \tau_i \in \mathcal{T}, \text{ and} \\ & y_{i,j} \geq 0, \forall \tau_i \in \mathcal{T}, \forall j = 1, 2, \cdots m'. \end{array} \tag{5b}$$

Our proposed algorithm, denoted as Algorithm ROUNDING, first derives a minimum feasible solution among the $2m$ equations of all combinations in Equation (5). Since Equations (5a) and (5b) are both standard linear programming problems, applying a polynomial-time linear programming solver, such as that in [9], could derive an optimal solution of Equation (5a) or Equation (5b) in polynomial time of n and m' for a fixed parameter m' if a feasible solution exists. If there does not exist any feasible solution for a fixed m' for Equation (5a) or Equation (5b), the infeasibility could also be determined in polynomial time. Since $\sum_{\tau_i \in \mathcal{T}} \min_{M_j \in \mathcal{M}} e_{i,j} \leq E_{budget}$, we could derive an optimal feasible solution $y^*_{i,j}$ for every $\tau_i \in \mathcal{T}$ and $j = 1, 2, \ldots, m'$ of Equation (5a) and Equation (5b) among $m' = 1, 2, \ldots, m$. For notational brevity, let $y^*_{i,j}$ be 0 for any $j > m'$, and y^* be the abbreviated vector for the variable assignment. Suppose that $\mathcal{T}_{I,j}$ is $\{\tau_i \in \mathcal{T} \mid y^*_{i,j} = 1\}$ for every $j = 1, 2, \ldots, m$ (Step 2). For the other tasks that are not in $\mathcal{T}_{I,j}$ for any $j = 1, 2, \ldots, m$, we denote these tasks as *fractional* tasks. Let $\mathcal{T}_F$ be the set of fractional tasks, i.e., $\mathcal{T}_F = \{\tau_i \in \mathcal{T} \mid \exists j \text{ with } 0 < y^*_{i,j} < 1\}$. In Steps 3 and 4, for each fractional task τ_i in $\mathcal{T}_F$, we insert τ_i to $\mathcal{T}_{I,j^*}$, where M_{j^*} is the processor type M_j with $y^*_{i,j} > 0$ and the minimum $E_{i,j}$. After the task assignment, in Steps 5 and 6, we then assign task set $\mathcal{T}_{I,j}$ by applying Algorithm FF to allocate processors of M_j and assign tasks onto the allocated processors of task set $\mathcal{T}_{I,j}$ for $j = 1, 2, \ldots, m$. After all, we just schedule these tasks by applying EDF individually on each allocated processor.

The energy consumption of the resulting solution is no more than $(\sum_{j=1}^{m} \sum_{\tau_i \in \mathcal{T}} E_{i,j} \cdot y^*_{i,j} \leq E_{budget})$, since we assign fractional tasks onto M_{j^*} which it is the processor type M_j with $y^*_{i,j} > 0$ and the minimum $E_{i,j}$. Hence, $\sum_{j=1}^{m} \sum_{\tau_i \in \mathcal{T}_{I,j}} E_{i,j} \leq E_{budget}$. Since applying Algorithm FF leads to an solution with energy consumption equal to $(\sum_{j=1}^{m} \sum_{\tau_i \in \mathcal{T}_{I,j}} E_{i,j})$, we know that the resulting solution of the MARTEC problem will not violate the energy constraint. The resulting solution executes each task on an allocated processor, and the total utilization of the tasks in each allocated processor is no more than 100%. Hence, applying EDF will make each task meet its deadline. Algorithm ROUNDING guarantees to derive a feasible solution of the MARTEC problem. The time complexity is $O(mP + n^2)$, where P is the time complexity for the applied linear programming solver. We now show the optimality of Algorithm ROUNDING in the following discussions.

As shown in [7], feasible solutions of a linear programming problem with γ variables form a *convex set* in γ dimensions. Moreover, an optimal solution for the linear programming equation is an *extreme point* of the convex set [7, 16], where an extreme point of the convex set is a member in the convex set which can not be expressed by the convex combination of any two distinct members in the convex set. We refer interested readers to [7, 16] for detailed definitions of convex sets and extreme points. In other words, an extreme point of the convex set of the feasible solutions for a linear programming problem with γ variables makes at least γ inequalities in the linear programming problem be tight. With such an observation of an optimal solution of the linear programming problem, we could show that there are at most two tasks in $\mathcal{T}_F$.

Algorithm 1 : ROUNDING

Input: $\mathcal{T}$, $\mathcal{M}$, and E_{budget};

1: let $y^*_{i,j}$ be the minimum feasible solution of Equation (5a) and Equation (5b) among $m' = 1, 2, \ldots, m$ for every $\tau_i \in \mathcal{T}$ and $j = 1, 2, \ldots, m'$;
2: $\mathcal{T}_{I,j} \leftarrow \{\tau_i \in \mathcal{T} \mid y^*_{i,j} = 1\}$ for every $j = 1, 2, \ldots, m$;
3: **for each** τ_i with $0 < y^*_{i,j} < 1$ for some $1 \leq j \leq m$ **do**
4: $\quad \mathcal{T}_{I,j^*} \leftarrow \mathcal{T}_{I,j^*} \cup \{\tau_i\}$, where M_{j^*} is the processor type M_j with $y^*_{i,j} > 0$ and the minimum $E_{i,j}$;
5: **for** $j \leftarrow 1$ to m **do**
6: $\quad$ allocate processors of M_j and assign the tasks in task set $\mathcal{T}_{I,j}$ by applying Algorithm FF;
7: return the EDF schedule of the resulting task assignment of $\mathcal{T}$ and processor allocation;

LEMMA 3. *If $\sum_{\tau_i \in \mathcal{T}} y^*_{i,m'} \cdot u_{i,m'}$ is 1, there are at most two tasks in $\mathcal{T}_F$; otherwise, $\mathcal{T}_F$ has only one task only.*

PROOF. There are $(n \cdot m' + n + 2)$ constraints and $n \cdot m'$ variables in both Equation (5a) and Equation (5b). Hence, the convex set of a solution of Equation (5a) or Equation (5b) is with $n \cdot m'$ dimensions. By the definition of the extreme point, there must be at least $n \cdot m'$ tight inequalities for an optimal solution of Equation (5a) or Equation (5b). If the first and second inequalities of the constraints in either Equation (5a) or Equation (5b) are tight in y^*, there must be at least $n \cdot m - 2 - n$ variable with $y^*_{i,j} = 0$. In other words, there are at most $2 + n$ variables with $y^*_{i,j} > 0$. Let β be the number of variables $y^*_{i,j}$ which is strictly between 0 and 1. δ denotes the number of variables $y^*_{i,j}$ equal to 1. We have

$$\begin{cases} \delta + \beta \leq n + 2, \text{and} \\ 2(n - \delta) \leq \beta, \end{cases} \tag{6}$$

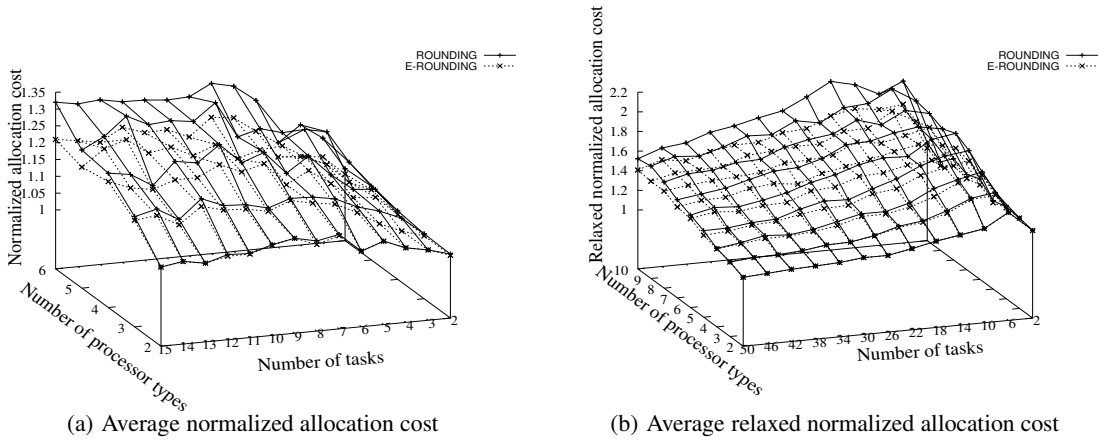
since there must be at least two variables strictly between 0 and 1 for a fractional task. By solving Equation (6), we can obtain two inequalities $\delta \geq n - 2$ and $\beta \leq 4$.

There are two cases: (1) $\delta = n - 2$, and (2) $\delta \geq n - 1$. When $\delta = n - 2$, β must be 4, and $\sum_{\tau_i \in \mathcal{T}} y^*_{i,m'} \cdot u_{i,m'}$ is 1. Otherwise, it is not a feasible solution of Equation (5). As a result, we have exactly two fractional tasks on this case. When $\delta \geq n - 1$, β is at most 3. Similarly, there is at most one fractional task. □

The following theorem states the approximation ratio of Algorithm ROUNDING.

THEOREM 3. *Algorithm* ROUNDING *is a polynomial-time $(m + 2)$-approximation algorithm for the* MARTEC *problem.*

PROOF. We prove this theorem by showing that the resulting solution of Algorithm ROUNDING would allocate at most $(m + 2)(\sum_{j=1}^{m'} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y^*_{i,j} \cdot C_j)$ when y^* comes from Equation (5a), or $(m + 2)(C_{m'} + \sum_{j=1}^{m'-1} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y^*_{i,j} \cdot C_j)$ when y^* comes from Equation (5b). Let $\hat{m}$ be the m', in which we derive y^*. By Lemma 1, we know that the resulting allocation of Algorithm



(a) Average normalized allocation cost (b) Average relaxed normalized allocation cost

Figure 1: The experimental result when (a) the number of available processor types ranged from 2 to 6, and the number of tasks ranged from 2 to 15; (b) the number of available processor types ranged from 2 to 10, and the number of tasks ranged from 2 to 50.

ROUNDING is at most $(\sum_{j=1}^{\hat{m}} \max\{1, 2\sum_{\tau_i \in \mathcal{T}_{I,j}} u_{i,j}\} C_j)$. Here are two cases, decided by whether $\sum_{\tau_i \in \mathcal{T}_{I,\hat{m}}} u_{i,\hat{m}} \cdot y^*_{i,\hat{m}} > 1$ or not, to be considered.

If $\sum_{\tau_i \in \mathcal{T}_{I,\hat{m}}} u_{i,\hat{m}} \cdot y^*_{i,\hat{m}} \leq 1$, i.e., an optimal solution comes from Equation (5b), the lower bound C^* of an optimal solution of the MARTEC problem would be $(C_{\hat{m}} + \sum_{j=1}^{\hat{m}-1} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y^*_{i,j} \cdot C_j)$. Let set $\mathcal{M}^\dagger$ be $\{M_j \in \mathcal{M} \mid \sum_{\tau_i \in \mathcal{T}_{I,j}} u_{i,j} \cdot y^*_{i,j} > 1\}$, and k is the cardinality of $\mathcal{M}^\dagger$. The allocation cost of processor types in $\mathcal{M}^\dagger$ to schedule tasks in $\cup_{M_j \in \mathcal{M}^\dagger} \mathcal{T}_{I,j} \setminus \mathcal{T}_F$ is at most $2\sum_{M_j \in \mathcal{M}^\dagger} \sum_{\tau_i \in \mathcal{T}_{I,j}} u_{i,j} \cdot y^*_{i,j} \cdot C_j \leq 2(C^* - C_{\hat{m}})$, and the number of processor types with $(\sum_{\tau_i \in \mathcal{T}_{I,j}} u_{i,j} \cdot y^*_{i,j} \leq 1)$ is $\hat{m} - k$. The total allocation cost of all fractional tasks can be bounded by $2C_{\hat{m}}$ because there are at most two tasks in $\mathcal{T}_F$. Therefore, the allocation cost of the solution derived by Algorithm ROUNDING is at most

$$2(C^* - C_{\hat{m}}) + (\hat{m} - k)C_{\hat{m}} + 2C_{\hat{m}} \leq (\hat{m} + 2)C^* \leq (m + 2)C^*,$$

since $C_j \leq C_{\hat{m}}$ for any $j < \hat{m}$.

If $\sum_{\tau_i \in \mathcal{T}_{I,\hat{m}}} u_{i,\hat{m}} \cdot y^*_{i,\hat{m}} > 1$, i.e., an optimal solution comes from Equation (5a) with at most one fractional task in $\mathcal{T}_F$ due to Lemma 3, the lower bound C^* of an optimal solution of the MARTEC problem would be $(\sum_{\tau_i \in \mathcal{T}_{I,\hat{m}}} u_{i,\hat{m}} \cdot y^*_{i,\hat{m}} \cdot C_{\hat{m}} + \sum_{j=1}^{\hat{m}-1} \sum_{\tau_i \in \mathcal{T}} u_{i,j} \cdot y^*_{i,j} \cdot C_j)$. Again, let k be the number of the set $\mathcal{M}^\dagger$. The allocation cost of the solution derived by Algorithm ROUNDING is at most

$$2(C^* - \eta C_{\hat{m}}) + (\hat{m} - k - 1)C_{\hat{m}} + 2\eta C_{\hat{m}} + C_{\hat{m}} \leq (m + 2)C^*,$$

where η is $\sum_{\tau_i \in \mathcal{T}_{I,\hat{m}}} u_{i,\hat{m}} \cdot y^*_{i,\hat{m}}$. A conclusion is reached. □

Another algorithm, denoted as Algorithm E-ROUNDING, to enhance the quality of the derived solutions of Algorithm ROUNDING could be done as follows: Algorithm E-ROUNDING selects a solution with the minimum resulting allocation cost by applying Steps 2 to 7 in Algorithm 1 among all of the feasible solutions of the $2m$ linear programming problem described in Equations (5a) and (5b) for $m' = 1, 2, \ldots, m$, by taking the solutions to derive $\mathcal{T}_{I,j}$ and $\mathcal{T}_F$ in Step 2 and Step 3 instead of y^*. Algorithm E-ROUNDING guarantees to derive a feasible solution of the MARTEC problem. The time complexity is $O(m(P + n^2))$, where P is the time complexity for the applied linear programming solver. Clearly, a solution of Algorithm E-ROUNDING is no worse than that of Algorithm ROUNDING for any input instance. Algorithm E-ROUNDING is a polynomial-time $(m + 2)$-approximation algorithm for the MARTEC problem.

We now show that the analysis is almost tight by providing a set of input instances with a gap close to m between the solutions of Algorithm ROUNDING and the optimal solutions. Suppose that we are given m processors so that $C_1 = \epsilon$, $C_2 = C_3 = \cdots = C_{m-1} = K - \epsilon$, and $C_m = K$, where $0 < \epsilon < K \neq \infty$. There are m tasks, $\tau_1, \tau_2, \ldots, \tau_m$. For τ_i in $\{\tau_2, \tau_3, \ldots, \tau_m\}$, $u_{i,1} = 1$, $E_{i,1} = E_{budget}/m$, $u_{i,i} = \epsilon/K - \delta$, $E_{i,i} = E_{budget}/m$, and $u_{i,j} = E_{i,j} = \infty$ for $j \neq i$ and $j > 1$, where $\epsilon/K > \delta > 0$. For τ_1, $u_{1,m} = 1$, $E_{1,m} = E_{budget}/m$, and $u_{1,j} = E_{1,j} = \infty$ for $j \neq m$. By applying Algorithm ROUNDING, $\mathcal{T}_{I,j}$ would consist of τ_j only, for $j = 2, 3, \ldots, m-1$. while $\mathcal{T}_{I,m}$ contains τ_1 and τ_m. The allocation cost of Algorithm ROUNDING for this input instance of the MARTEC problem is $2K + (K - \epsilon)(m - 2)$, whereas the optimal solution is $K + (m - 1)\epsilon$. Clearly, when ϵ is small, any solution derived by Algorithm ROUNDING is with a gap m to an optimal one.

4. PERFORMANCE SIMULATION

4.1 Experimental Setups

In this section, we provide extensive evaluations for the proposed algorithms. The hyper-period was 1000 ms. The number of jobs of task τ_i within the hyper-period, denoted by t_i, was an integral integer uniformly distributed in the range of $[1, 100]$. Hence, the period of task τ_i was set as $\frac{1000}{t_i}$ ms. The execution time $c_{i,j}$ of jobs of task τ_i on processor type M_j was a random variable uniformly distributed in the range of $[1, \frac{1000}{t_i}]$, and the energy consumption $e_{i,j}$ of jobs of task τ_i on processor type M_j was a random variable in the range of [100, 1000]. For each processor type M_j, the cost C_j was an integral variable uniformly distributed in the range of $[100, 1000]$. For a specified energy budget ratio f, the energy budget for a given task set $\mathcal{T}$ on a set of processor types $\mathcal{M}$ was set as $(E_{max} - E_{min})f + E_{min}$, where $E_{min} = \sum_{\tau_i \in \mathcal{T}} \min_{M_j \in \mathcal{M}} e_{i,j}$ and $E_{max} = \sum_{\tau_i \in \mathcal{T}} \max_{M_j \in \mathcal{M}} e_{i,j}$.

We run two types of experiments by first varying the number of processor types and the number of tasks and then adjusting the energy budget. For the first type, two different configurations were experimented by setting the energy budget ratio as 0.1. For the first configuration, the numbers of processor types and tasks were from 2 to 6 and 2 to 15, respectively. For the second configuration, the numbers of processor types and tasks were from 2 to 10 and 5 to 50, respectively. For the second type, we varied the energy budget ratio from 0.05 to 1, stepped by 0.05, where the number of pro-

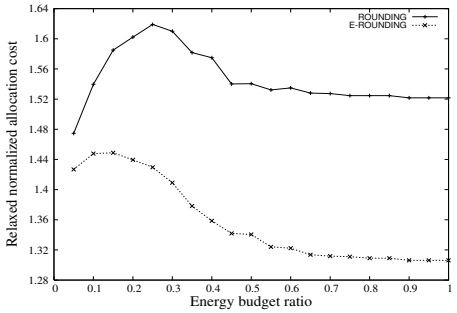


Figure 2: The average relaxed normalized allocation cost when the energy budget ratio ranged from 0.05 to 1.

cessor types is an integral variable in the range of $[2, 10]$, and the number of tasks is an integral variable in the range of $[2, 50]$. Each configuration was evaluated with 128 independent settings.

The *normalized allocation cost* was adopted as the performance metric in the experiments. The normalized allocation cost of an algorithm for an input instance is the ratio of the allocation cost of the solution derived from the algorithm to that of the optimal solution by an exhaustive search. The *relaxed normalized allocation cost* of an algorithm for an input instance is the ratio of the allocation cost of a solution derived from the algorithm to that of the lower bound derived from the minimum cost among the $2m$ linear programming problems of Equation (5).

4.2 Experimental Results

Figure 1(a) shows the average normalized allocation cost of Algorithms ROUNDING and E-ROUNDING when the energy budget ratio was 0.1, the number of processor types varied from 2 to 6, and the number of tasks varied from 2 to 15. Figure 1(b) shows the average relaxed normalized allocation cost of Algorithms ROUNDING and E-ROUNDING, when the energy budget ratio was 0.1, the number of processor types varied from 2 to 10, and the number of tasks varied from 5 to 50 stepped by 5. The performance of E-ROUNDING was no worse than that of ROUNDING in the experimental results. Both of the proposed algorithms could derive solutions with costs close to those of optimal solutions. The performance gap between the two algorithms became wider for a larger number of processor types. In Figure 1(a) (/Figure 1(b)), the average normalized allocation cost (/relaxed normalized allocation cost) became larger when the number of processor types increased. However, the growing tendency in both figures was slow. The maximum values of the average relaxed normalized allocation costs of ROUNDING and E-ROUNDING in Figure 1(b) were 2.118 and 1.866, respectively.

Figure 2 shows the average relaxed normalized allocation cost of Algorithms ROUNDING and E-ROUNDING when the energy budget ratio varied from 0.05 to 1 with a step equal to 0.05. The average relaxed normalized allocation cost was almost the same when the energy budget ratio was greater than 0.65, since the energy constraint was easier to satisfy when the energy budget ratio was larger. Again, both of the proposed algorithms could derive solutions with costs close to those of optimal ones. Moreover, Algorithm E-ROUNDING outperformed Algorithm ROUNDING, especially when the energy budget ratio was large.

5. CONCLUSION

This paper targets energy-efficient scheduling of periodic hard real-time tasks with an objective to minimize the total cost of processors under a given energy constraint. The problem is $\mathcal{NP}$-hard even when the number of the available processor types is a constant. It is also shown that there does not exist any polynomial-time algorithm with a constant approximation ratio, unless $\mathcal{NP} = \mathcal{P}$. We first propose an approximation algorithm based on a rounding technique with the approximation ratio $(m + 2)$, where m is the number of the available processor types. The algorithm is then improved with better solutions in many cases. The performance of the proposed algorithms was evaluated by a series of experiments, compared to optimal solutions. Experimental results show that the proposed algorithms could always derive solutions with costs close to those of optimal solutions.

With the rapid evolving of voltage-scaling technology, the results in this paper should be further extended with dynamic voltage scaling. More researches in this direction might be rewarding in the future.

References

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

[2] 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.

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

[4] S. K. Baruah. Partitioning real-time tasks among heterogeneous multiprocessors. In *Proceedings of the 33rd International Conference on Parallel Processing*, pages 467–474, 2004.

[5] 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.

[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] G. B. Dantzig and M. N. Thapa. *Linear Programming 1: Introduction*. Springer Verlag, 1997.

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

[9] GNU Linear Programming Kit. http://www.gnu.org/software/glpk/glpk.html.

[10] N. K. Jha. Low power system scheduling and synthesis. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 259–263, 2001.

[11] D. Kirovski and M. Potkonjak. System-level synthesis of low-power hard real-time systems. In *Proceedings of the 34th ACM/IEEE Conference on Design Automation Conference*, pages 697–702, 1997.

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

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

[14] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[15] Z. Shao, Q. Zhuge, X. Chun, and E. H.-M. Sha. Efficient assignment and scheduling for heterogeneous dsp systems. *IEEE Transaction on Parallel and Distributed Systems*, 16(6):516–525, June 2005.

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

[17] 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.

[18] 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.