# (1+$\epsilon$) Approximation Clock Rate Assignment for Periodic Real-Time Tasks on A Voltage-Scaling Processor

Jian-Jia Chen
Department of Computer Science and Information Engineering
National Taiwan University, Taiwan
r90079@csie.ntu.edu.tw

Tei-Wei Kuo and Chi-Sheng Shih
Department of Computer Science and Information Engineering
Graduate Institute of Networking and Multimedia
National Taiwan University, Taiwan
{ktw,cshih}@csie.ntu.edu.tw

## ABSTRACT

Energy-efficient scheduling is an effective way to balance the system performance and the energy consumption. We design a polynomial-time $(1+\epsilon)$-approximation algorithm to minimize the energy consumption for periodic real-time tasks over such processors, where $\epsilon$ is the tolerable error given by users ($1 \geq \epsilon > 0$). It provides trade-offs between the user's tolerable error and the runtime complexity including the time complexity and the memory space complexity. System engineers could trade performance with implementation constraints.

**Categories and Subject Descriptors:** C.3 [SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS] - Real-Time and Embedded Systems

**General Terms:** Algorithms, Design.

**Keywords:** Energy-Efficient Scheduling, DVS Scheduling, Real-Time Systems, Energy Consumption Minimization.

## 1. INTRODUCTION

With the advance technology of VLSI circuit designs, many modern processors can operate at various (processor) clock rates with negligible voltage switching overheads. The lower the clock rate is, the less the power consumption is. To cut the power bills or prolong the lifetime of battery-powered embedded systems, it is desirable to reduce the energy consumption as much as possible. However, a task executing on a processor with a lower clock rate usually needs more execution time. Hence, reducing the clock rate can certainly reduce the energy consumption but may lead to the violation of the timing requirements of the systems.

In this paper, we are interested in the clock rate assignment problem for periodic real-time task scheduling when a processor has a finite number of available clock rates, in which all job instances of a real-time task are executed at the same clock rate. We shall not only guarantee the schedulability of tasks but also minimize the energy consumption. Our work is closely related to the work by Mejía-Alvarez et al. [4], in which the total energy saving was pursued, compared to task executions at the highest clock rate. Both of the algorithms proposed in [4] could be applied to the problem concerned in this paper. However, the approximation ratios of the proposed algorithms in [4] are not constants for the minimization of energy consumption. In this paper, we propose a polynomial-time $(1+\epsilon)$-approximation algorithm, where $\epsilon$ is the tolerable error margin given by users ($1 \geq \epsilon > 0$). We show that the energy consumption of any derived solution would not be more than $(1+\epsilon)$ times of an optimal solution. The proposed algorithm allows the users to trade the system performance with implementation constraints. When there is not enough time to compute the near optimal solution, the users can specify a large tolerable error margin $\epsilon$. When there is enough time to compute, the users can specify a small tolerable error margin $\epsilon$ to obtain a near optimal solution. The performance of the proposed algorithm is evaluated by a series of experiments, compared to an exponential algorithm based on an exhaustive search and algorithms proposed in [4].

The rest of this paper is organized as follows: In Section 2, we define the system models and formulate the problem. Our proposed approximation algorithm for minimizing energy consumption is then presented in Section 3. Experimental results are shown in Section 4. Section 5 concludes this paper.

## 2. MODELS AND PROBLEM DEFINITION

Thus far, and in our subsequent discussion, we use the terms task and job as they are commonly used in real-time systems literature [2, 3, 5]. A *job* is an instance of signal processing, computation, and so on. A *task* is a sequence of jobs. We call tasks $T_1$, $T_2$, etc. A task set is denoted by $\mathbf{T} = \{T_1, T_2, \ldots, T_N\}$ where $N$ is the number of tasks in set $\mathbf{T}$.

We assume that the processor can operate at $M$ distinctive clock rates where $M$ is an integer and no less than 2. The set of available clock rates of a processor is denoted by $\mathbf{F} = \{f_1, f_2, ..., f_M\}$ in which clock rates are indexed in an ascendent order. The *arrival time* of a task, denoted by $a$, is the instant of time at which the task becomes known to the scheduler. The *execution CPU cycles* of a task, denoted by $c$, is the number of CPU cycles required to complete the execution of any job of the task when it executes alone and has all the resources it requires. The *period* of a task, denoted by $p$, is the minimal arrival interval between two consecutive jobs of the task. In this paper, the relative deadline of a task is assumed to be equal to the period of the task. Hence, a task $T_i$ is defined as $T_i = (a_i, c_i, p_i)$. The hyper-period for task set $\mathbf{T}$, denoted by $L$, is the least common multiple (LCM) of the periods of the tasks in task set $\mathbf{T}$.

The *assigned clock rate* for a task $T_i$, denoted by $r_i$, is the clock rate of the processor when task $T_i$ executes and is one of the available clock rates. The *clock rate assignment* for set $\mathbf{T}$, denoted by $R$, is a vector of assigned clock rates for all the tasks $T_i$ in $\mathbf{T}$, in which $R = (r_1, r_2, ..., r_N)$ and $r_i \in \{f_1, f_2, \ldots, f_M\}$ for $1 \leq i \leq N$. When we are concerned with the clock rate assignment for task subset $\{T_1, T_2, \ldots, T_i\}$ where $i \leq N$, *partial clock rate assignment* $R_i$ denotes the clock rate assignment for the subset.

The *power consumption function* for task $T_i$ is denoted by $P_i(r_i)$ where $r_i$ is the assigned clock rate for task $T_i$. Without loss of generality, we assume that power consumption functions are strictly

*This work is supported in part by a grant from the NSC program 93-2752-E-002-008-PAE and in part by a grant from the NSC program 93-2213-E-002-090.


*EMSOFT'05*, September 19–22, 2005, Jersey City, New Jersey, USA.

convex and increasing functions, e.g., $P_i(s) \propto s^3$ [4, 7]. Similar to most earlier works, we assume that the execution time of a job is proportional to the clock rate chosen to execute the job. When the assigned clock rate for task $T_i$ is $f_j$, the instantaneous utilization of task $T_i$ is $U_{ij} = \frac{c_i}{p_i} \cdot \frac{1}{f_j}$. The *energy consumption* for $T_i$ executed at $r_i$ for the hyper-period is denoted by $\psi(T_i, r_i)$, where $\psi(T_i, r_i) = \frac{L}{p_i}(P_i(r_i)\frac{c_i}{r_i})$. For task set $\mathbf{T}$ and its corresponding clock rate assignment $R$, the total energy consumption, denoted by $\Psi(\mathbf{T}, R)$, is the sum of energy consumption of all of the tasks in set $\mathbf{T}$, i.e., $\Psi(\mathbf{T}, R) = \sum_{1 \le i \le N} \psi(T_i, r_i)$.

We also assume that tasks are preemptible and independent. Liu and Layland [2] show that the earliest-deadline-first (EDF) policy is the optimal scheduling algorithm on a uniprocessor environment. A task set scheduled by the EDF policy is *feasible* if and only if the total instantaneous utilization of the task set is no more than 1 [2]. Hence, a clock rate assignment $R$ for set $\mathbf{T}$ is *feasible* if and only if $\sum_{1 \le i \le N}(\frac{c_i}{p_i} \cdot \frac{1}{r_i}) \le 1$. Furthermore, a partial clock rate assignment $R_i$ is feasible if and only if $\sum_{k=1}^{i}(\frac{c_k}{p_k} \cdot \frac{1}{r_k}) \le 1$. A feasible clock rate assignment, denoted by $R^*$, is *optimal* in the sense that its energy consumption is the minimum among all feasible clock rate assignments.

The MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT problem is to find a feasible clock rate assignment $R$ for task set $\mathbf{T}$ so that the sum of energy consumption for all the tasks in set $\mathbf{T}$ is minimized. Theorem 1 shows that the MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT problem is $\mathcal{NP}$-hard.

THEOREM 1. *The* MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT *problem is $\mathcal{NP}$-hard.*

PROOF. The theorem can be proved by a reduction from the $\mathcal{NP}$-complete SUBSET SUM problem [1] and, hence, the details are omitted due to the space limitation. □

## 3. OUR APPROXIMATION ALGORITHM

The proposed algorithm, denoted as Algorithm $\epsilon$-ME, is a dynamic programming algorithm. The rationale of the algorithm is to first group the possible energy consumptions to a limited number of energy consumption groups. The representive energy consumption of each group is called the *rounded-up energy consumption*. It is because every possible energy consumption in an energy consumption group is no greater than the representive energy consumption. Instead of finding the optimal clock rate assignment $R^*$, the algorithm chooses a feasible clock rate assignment for $\mathbf{T}$ such that the sum of the rounded-up energy consumption for task set $\mathbf{T}$ is minimized. We will show that the algorithm completes in polynomial time and the energy consumption for the chosen clock rate assignment is no more than $(1 + \epsilon)$ times of the optimal energy consumption where $\epsilon$ is a design parameter.

For the sake of brevity, we define ratio $\gamma$ as

$$\gamma = \max_{1 \le i \le N, 2 \le j \le M}\{\frac{\psi(T_i, f_j)}{\psi(T_i, f_{j-1})}, 1\} = \max_{1 \le i \le N, 2 \le j \le M}\{\frac{P_i(f_j)f_{j-1}}{P_i(f_{j-1})f_j}, 1\}.$$

In other words, the energy consumption to execute a task at available clock rate $f_j$ is at most $\gamma$ times of that at the clock rate $f_{j-1}$ for any $2 \le j \le M$. Algorithms SGA and EGA proposed in [4] can also be applied to solve the MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT problem. The following theorem shows that the two algorithms provide a $\gamma$-approximation ratio.

LEMMA 1. *When Algorithms SGA and EGA proposed in [4] are applied to the* MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT *problem, the approximation ratio is $\gamma$, which is tight.*

PROOF. It is not difficult to show the approximation ratio is no greater than $\gamma$ since there is at most one task assigned with two clock rates $f_j$ and $f_{j+1}$ in the optimal solution of the problem (P2) in [4]. We show the tightness by considering the following input instance of $\mathbf{T}$ with $N$ tasks: $T_1 = (0, p(1 - (N-1)\lambda/2), p)$, $P_1(f) = f^3$, $T_2 = (0, p \cdot \lambda, p)$, $T_N = T_{N-1} = \cdots = T_3 = T_2$, $P_2(f) = P_3(f) = \cdots = P_N(f) = (1 + \delta)f^3$, and $(f_1, f_2, f_3, \ldots, f_M) = (1, \rho, \rho^2, \ldots \rho^{M-1})$, where $\delta > 0$, $\rho \ge 2$, $M \ge 3$, and $(1+\delta)(N-1)\lambda\rho^{(2M-4)} = (1 - (N-1)\lambda/2)$. It is clear that $\gamma = \rho^2$ and $\frac{1}{4(N-1)} > \lambda > 0$. Both algorithms SGA and EGA derive a clock rate assignment $R_A$ to execute $T_1$ at clock rate $f_2$ and the other tasks at clock rate $f_1$. The energy consumption of such a clock rate assignment is $p((1 + \delta)(N-1)\lambda + (1 - (N-1)\lambda/2)\rho^2)$. Because $1-(N-1)\lambda/2+(N-1)\lambda/\rho < 1$, executing $T_1$ at clock rate $f_1$ and the others at clock rate $f_2$ is also a feasible clock rate assignment, and the energy consumption is $p((1+\delta)(N-1)\lambda\rho^2+(1-(N-1)\lambda/2))$. Since $(1 + \delta)(N-1)\lambda\rho^{(2M-4)} = (1 - (N-1)\lambda/2)$, we have

$$\frac{\Psi(\mathbf{T}, R_A)}{\Psi(\mathbf{T}, R^*)} \ge \frac{p((1 + \delta)(N-1)\lambda + (1 - (N-1)\lambda/2)\rho^2)}{p((1 + \delta)(N-1)\lambda\rho^2 + (1 - (N-1)\lambda/2))} \approx \rho^2 = \gamma.$$

□

For the rest of this section, we show how to exploit the clock rate assignment derived from Algorithm SGA [4] to $(1+\epsilon)$-approximation solutions. Before we describe our proposed Algorithm $\epsilon$-ME, we define several terms used for the algorithm. *Rounding factor*, denoted by $q$, for all tasks in the task set $\mathbf{T}$ is a positive real. Rounding factor $q$ determines the granularity of rounded-up energy consumptions, which is the inverse of the rounding factor, i.e., $1/q$. Hence, the less the rounding factor $q$ is, the greater rounding error of the rounded-up energy consumption is. However, the less rounding factors $q$ is, the higher the runtime overhead the algorithm has. The *rounded-up energy consumption* for executing task $T_i$ at clock rate $f_j$ for rounding factor $q$, denoted by $\psi_q(T_i, f_j)$, is $\psi_q(T_i, f_j) = \frac{\lceil q \cdot \psi(T_i, f_j) \rceil}{q}$. Naturally, when the rounding factor is $q$ and the feasible partial clock rate assignment is $R_i$ $(1 \le i \le N)$, the rounded-up energy consumption for $R_i$ is $\Psi_q(\mathbf{T}, R_i) = \sum_{m=1}^{i} \psi_q(T_m, r_m)$. If we can have a good control on the rounding error of the energy consumption, we can have a good approximated solution and acceptable runtime overhead. We will show how to choose a good rounding factor $q$ later. From now on, we will use term $k$ to denote the amount of rounded-up energy consumption in the scale of $1/q$ energy unit where $k$ is a positive integer.

The *utilization lower bound* for task subset $\{T_1, \ldots, T_i\}$, denoted by $B_q(i, k)$, is the minimum total utilization for all feasible partial clock rate assignments $R_i$s such that the total rounded-up energy consumption of the task set is no greater than $k$ units of rounded-up energy, i.e., $\Psi_q(\mathbf{T}, R_i) \le k/q$. If there exists no feasible partial clock rate assignment $R_i$ such that $\Psi_q(\mathbf{T}, R_i)$ is also no greater than $k/q$, then $B_q(i, k)$ is undefined. In the following, we show how to derive $B_q(i, k)$ from $B_q(i-1, \cdot)$. If there exist clock rates $f_j$s such that $B_q(i-1, k - q \cdot \psi_q(T_i, f_j))$ is defined and $B_q(i-1, k - q \cdot \psi_q(T_i, f_j)) + \frac{c_i}{p_i f_j}$ is no greater than 1, then $B_q(i, k)$ is defined and the value of $B_q(i, k)$ is the minimum $B_q(i-1, k-q\cdot\psi_q(T_i, f_j)) + \frac{c_i}{p_i f_j}$. Otherwise, $B_q(i, k)$ is undefined.

To compute the utilization lower bound, we define the boundary conditions of $B_q(i, k)$ as follows.

$$B_q(i, k) = \begin{cases} 0 & \text{if } k \ge 0 \text{ and } i = 0; \\ \infty\text{(i.e., undefined)} & \text{if } k < 0 \text{ and } 0 \le i \le N. \end{cases} \quad (1)$$

Therefore, the following recursive relation holds for any $1 \le i \le N$ and any non-negative integer $k$:

$$B_q(i, k) = \min_{j=1,\ldots,M} \begin{cases} B_{ijk} & \text{if } B_{ijk} \le 1; \\ \infty\text{(i.e., undefined)} & \text{otherwise,} \end{cases} \quad (2)$$

where $B_{ijk} = B_q(i-1, k-q\cdot\psi_q(T_i, f_j)) + \frac{c_i}{p_i f_j}$ for abbreviations. When the utilization lower bound for set $\mathbf{T}$ is defined and $k_{q,\min}$ is the minimum integer among all possible $k$s, we call $k_{q,\min}/q$ the *minimum rounded-up energy consumption* for rounding factor $q$; that is, $\Psi_q(\mathbf{T}, R)$ is no less than $k_{q,\min}/q$ for any feasible clock rate assignment $R$ for $\mathbf{T}$.

**Algorithm 1** : $\epsilon$-ME

**Input:** $(\mathbf{T}, \{f_1, \ldots, f_M\}, \epsilon)$;
**Output:** A feasible clock rate assignment for $\mathbf{T}$;
1: **if** $\sum_{i=1}^{N} \frac{c_i}{p_i \cdot f_M} > 1$ **then**
2: return *no feasible clock rate assignment exists*;
3: obtain $\hat{R}$ by calling Algorithm SGA proposed in [4] in $O(MN \log MN)$ time;
4: $q \leftarrow N/(\epsilon \cdot \Psi(\mathbf{T}, \hat{R}))$;
5: **while** true **do**
6: $\psi_q(T_i, f_j) \leftarrow \frac{\lceil q \cdot \psi(T_i, f_j) \rceil}{q}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M$;
7: $k \leftarrow 0$
8: **while** true **do**
9: **for** $i \leftarrow 1$ to $N$ **do**
10: compute $B_q(i, k)$ according to Equations (1) and (2);
11: **if** $B_q(N, k) \neq \infty$ **then**
12: break;
13: **else**
14: $k \leftarrow k + 1$;
15: **if** $\epsilon \cdot k \geq 2N$ **then**
16: break;
17: **else**
18: $q \leftarrow q * 2$;
19: backtrack the dynamic table generated by lines 5-18; let $R^q$ be the resulting clock rate assignment;
20: return $R^q$;

The pseudo-code of Algorithm $\epsilon$-ME is shown in Algorithm 1. The goal of our algorithm is to define the utilization lower bound for each rounded-up energy consumption, i.e., to determine a feasible clock rate assignment. Since $q \cdot \psi_q(T_i, f_j)$ is a non-negative integer, the utilization lower bound $B_q(i, k)$ can be obtained by constructing a dynamic programming table. The algorithm starts from the case that the task set has only one task $T_1$ and variable $k$ is 0; the initial value of rounding factor $q$ is $N/(\epsilon \cdot \Psi(\mathbf{T}, \hat{R}))$, where $\epsilon$ is a user input parameter and $0 < \epsilon \leq 1$ and $\hat{R}$ is the clock rate assignment derived from Algorithm SGA [4]. From Lines 6 to 14, the algorithm constructs a dynamic programming table by applying Equations (1) and (2) until $B_q(N, k)$ is defined. After $B_q(N, k)$ is defined at Line 11, the algorithm doubles the rounding factor $q$ to refine the solution. For each iteration, the algorithm repeats the dynamic programming until the termination condition satisfies, i.e., $\epsilon \cdot k_{q,\min} \geq 2N$, given at Line 15. When the termination condition satisfies, a feasible clock rate assignment can be derived by backtracking the dynamic programming table built in the earlier steps.

We now prove the optimality of the proposed algorithm. For the rest of this section, let $R^q$ be a feasible clock rate assignment for $\mathbf{T}$ such that $\Psi_q(\mathbf{T}, R^q) = k_{q,\min}/q$. By the definition of $k_{q,\min}$, the rounded-up energy consumption of any feasible clock rate assignment for $\mathbf{T}$ is no less than $k_{q,\min}/q$. Therefore, we have

$$\Psi_q(\mathbf{T}, R^q) \leq \Psi_q(\mathbf{T}, R), \tag{3}$$

for any feasible clock rate assignment $R$ of $\mathbf{T}$ and any rounding factor $q$. Since $\psi_q(T_i, f_j)$ is defined as $\frac{\lceil q \cdot \psi(T_i, f_j) \rceil}{q}$, we know that $\psi(T_i, f_j) \leq \psi_q(T_i, f_j) \leq \psi(T_i, f_j) + \frac{1}{q}$. Therefore, the following inequality

$$\Psi(\mathbf{T}, R) \leq \Psi_q(\mathbf{T}, R) \leq \Psi(\mathbf{T}, R) + \frac{N}{q} \tag{4}$$

holds for any clock rate assignment $R$ of $\mathbf{T}$ due to the aggregation error of these $N$ tasks. By the optimality of $R^*$, and Equations (3) and (4), we have

$$\begin{aligned} \Psi(\mathbf{T}, R^*) &\leq \Psi(\mathbf{T}, R^q) \leq \Psi_q(\mathbf{T}, R^q) \\ &\leq \Psi_q(\mathbf{T}, R^*) \leq \Psi(\mathbf{T}, R^*) + \tfrac{N}{q}. \end{aligned} \tag{5}$$

We shall show that the condition in Line 15 in Algorithm $\epsilon$-ME, i.e., $\epsilon \cdot k_{q,\min} \geq 2N$, leads us to have a $(1+\epsilon)$-approximation solution.

LEMMA 2. *If $\epsilon \cdot k_{q,\min} \geq 2N$ and $0 < \epsilon \leq 1$, then the energy consumption for the corresponding clock rate assignment $R^q$ of $\mathbf{T}$ is no greater than $(1+\epsilon)$ times of the minimum energy consumption, i.e., $\Psi(\mathbf{T}, R^q) \leq (1+\epsilon)\Psi(\mathbf{T}, R^*)$.*

PROOF. The inequality of $\epsilon \cdot k_{q,\min} \geq 2N$ implies $q \geq \frac{2N}{\epsilon \cdot \Psi_q(\mathbf{T}, R^q)}$. Therefore, combining with Equation (5), we have $\frac{N}{q} \leq \frac{\epsilon \cdot \Psi_q(\mathbf{T}, R^q)}{2} \leq \frac{\epsilon \cdot \Psi(\mathbf{T}, R^*)}{2} + \frac{\epsilon \cdot N}{2q}$. With $0 < \epsilon \leq 1$, it follows that $\frac{N}{q} \leq \frac{\epsilon \cdot \Psi(\mathbf{T}, R^*)}{2-\epsilon} \leq \epsilon \cdot \Psi(\mathbf{T}, R^*)$. By Equation (5), we have $\Psi(\mathbf{T}, R^q) \leq (1+\epsilon) \cdot \Psi(\mathbf{T}, R^*)$. □

It remains to show that the time complexity and the space complexity are both polynomial in the input size and $\frac{1}{\epsilon}$.

LEMMA 3. *Algorithm $\epsilon$-ME takes $O(N^2 M(\epsilon^{-1} + \log \gamma))$ time and $O(\epsilon^{-1} N^2)$ space for any parameter $0 < \epsilon \leq 1$,*

PROOF. For notational brevity, let $q'$ be the initial rounding factor $q$ and $q''$ be the termination rounding factor in Algorithm $\epsilon$-ME. That is, $q' = \frac{N}{\epsilon \cdot \Psi(\mathbf{T}, \hat{R})}$ and $\epsilon \cdot k_{q'',\min} \geq 2N$. For a rounding factor $q$, building a dynamic programming table $B_q()$ until $B_q(N, k_{q,\min})$ is defined takes

$$O(MN \cdot k_{q,\min}) = O(MN \cdot \Psi_q(\mathbf{T}, R^q) \cdot q) \tag{6}$$

time and $O(N \cdot \Psi_q(\mathbf{T}, R^q) \cdot q)$ space. If the initial rounding factor is equal to the termination rounding factor, Algorithm $\epsilon$-ME takes $O(N^2 M/\epsilon)$ time and $O(N^2/\epsilon)$ space since

$$\begin{aligned} O(\Psi_{q'}(\mathbf{T}, R^{q'}) \cdot q') &= O(\Psi(\mathbf{T}, R^*) \cdot q' + N) \\ &= O(\tfrac{N\Psi(\mathbf{T}, R^*)}{\epsilon \Psi(\mathbf{T}, \hat{R})} + N) = O(\tfrac{N}{\epsilon}), \end{aligned}$$

where the first equality comes from Equation (5). In the following, we focus on the other case that $q'' \neq q'$. By Equations (5) and (6), we know that each iteration for a specified $q$ to build a dynamic programming table takes $O(MN(q \cdot \Psi(\mathbf{T}, R^*) + N))$ time. Similarly, the space complexity is $O(N(q \cdot \Psi(\mathbf{T}, R^*) + N))$. Since the rounding factor $q$ is doubled in each iteration, the overall running time is

$$O\left(MN\left(q'' \cdot \Psi(\mathbf{T}, R^*) + N \log \frac{q''}{q'}\right)\right),$$

and the overall space is

$$O\left(N\left(q'' \cdot \Psi(\mathbf{T}, R^*) + N\right)\right). \tag{7}$$

Furthermore, we have

$$\epsilon \cdot k_{q''/2,\min} < 2N \Rightarrow \frac{q''}{2} < \frac{2N}{\epsilon \cdot \Psi_{q''/2}(\mathbf{T}, R^{q''/2})}, \tag{8}$$

for the $(\log_2 \frac{q''}{q'} - 1)$-th iteration. By Equation (5), we have

$$q'' < \frac{4N}{\epsilon \cdot \Psi_{q''/2}(\mathbf{T}, R^{q''/2})} \leq \frac{4N}{\epsilon \cdot \Psi(\mathbf{T}, R^*)}. \tag{9}$$

Therefore, we know

$$q'' \cdot \Psi(\mathbf{T}, R^*) = O(N/\epsilon),$$

and the space complexity is $O(\epsilon^{-1} N^2)$. Combining the known relation of $\Psi(\mathbf{T}, R^*)$ and $\Psi(\mathbf{T}, \hat{R})$ and Equation (9), we have

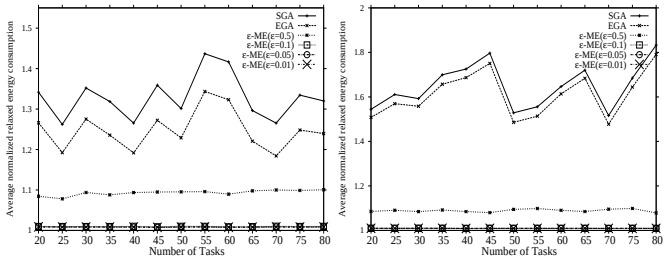
$$\log_2 \frac{q''}{q'} < \log_2 \frac{4\Psi(\mathbf{T}, \hat{R})}{\Psi(\mathbf{T}, R^*)} = O(\log \gamma).$$

Therefore, the time complexity is $O(N^2 M(\epsilon^{-1} + \log \gamma))$. □

After all, since $\log \gamma$ is polynomial in the number of bits required to encode the power consumption functions $P_i(f_j)$ for all $1 \leq i \leq N$ and $1 \leq j \leq M$, we have the following theorem by combining the above lemmas.

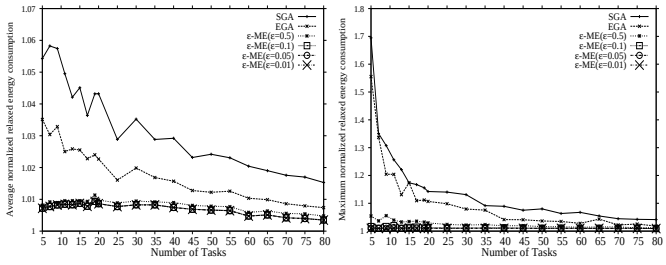
THEOREM 2. *Algorithm $\epsilon$-ME is a fully polynomial-time approximation scheme of the* MIN ENERGY CONSUMPTION CLOCK RATE ASSIGNMENT *problem to compute a solution no greater than $(1+\epsilon)$-optimal solution for any user-specified parameter $\epsilon$, where $0 < \epsilon \leq 1$.*

PROOF. This comes directly from Lemmas 1, 2, and 3. □




(a) Type-I (b) Type-II

**Figure 1: The average normalized relaxed energy consumption for the distribution of Type-I and II.**




(a) (b)

**Figure 2: The average and maximum normalized relaxed energy consumptions for the distribution of Type-III.**

## 4. EXPERIMENTAL RESULTS

The performance of Algorithm $\epsilon$-ME is evaluated by extensive evaluations. We compare the performance of Algorithm $\epsilon$-ME with Algorithm SGA and Algorithm EGA [4]. The $95\%$ confidence intervals for the data sets in the results are no more than $5\%$ of their data values for $256$ independent runs on each configuration. The proposed algorithm is evaluated for task sets with various workload. The workdload are generated based on two parameters: utilization and number of jobs within time interval $L$. Tasks in Type-I are first randomly determined whether they are tasks with high or low utilization. Tasks have a probability equal to $(1 - (2/N))$ being a task with low utilization, where $N$ is the number of tasks in each experimented task set. For tasks with low utilizations, the maximal instantaneous utilization of such a task $T_i$ (i.e., the utilization of $T_i$ when the system operated at the lowest clock rate) is randomly chosen in the range $(0, 1/(5N)]$. For tasks with high utilizations, the maximal instantaneous utilization of such a task is randomly chosen in the range $[1/(5N), 1]$. Task sets in Type-II have one task with high (maximal instantaneous) utilization randomly selected between $0.9$ and $1.1$. The maximal instantaneous utilizations of other tasks are randomly selected between $1/(10N)$ and $1/(5N)$. In Type-III, the maximal instantaneous utilization of each task is randomly selected between $1/(2N)$ and $2/N$.

The number of job instances of task $T_i$ within the hyper-period $L$, denoted by $b_i$, is an integral variable uniformly distributed in the range $[1, 16]$. The hyper-period $L$ of the tasks is set as $32,000$ units of time. Then, the worst-case CPU execution cycles $c_i$ for task $T_i$ is $c_i = f_1 U_{i1} p_i$. The power consumption function of task $T_i$ is $P_i(f) = \alpha_i f^3$ for $1 \leq i \leq N$, where the value of $\alpha_i$ is uniformly distributed in the range of $[2, 10]$. The target processor is the Intel XScale with five clock rates, i.e., 150MHz, 400MHz, 600MHz, 800MHz, and 1000MHz [6]. The evaluations are performed by normalizing the clock rates. Therefore, $M = 5$ and $(f_1, f_2, \ldots, f_5) = (0.15, 0.4, 0.6, 0.8, 1.0)$.

The *normalized relaxed energy consumption* of an algorithm for an input instance is defined as the ratio of the energy consumption for the clock rate assignment derived from the algorithm to the lower bound of the input instance by solving problem P2 in [4].

Due to the space limit, only representative results are shown. Figure 1 shows the performance for the case when there are five distinctive clock rates for the distributions of Type-I and Type-II, where $(f_1, f_2, \ldots, f_5) = (0.15, 0.4, 0.6, 0.8, 1.0)$. Figures 1(a) and 1(b) show the average normalized relaxed energy consumption for Type-I and II workload when the number of tasks range from 20 to 80 stepped by 5, respectively. Both of the clock rate assignments derived from Algorithms SGA and EGA have worse performance than those of Algorithm $\epsilon$-ME. The performance of Algorithm $\epsilon$-ME is steady regardless of the number of tasks in $\mathbf{T}$ for the either Type-I or Type-II workload. Algorithms SGA and EGA perform not well for Type-II workload due to the possibility of improper clock rate assignment of the large task (i.e., the task whose maximal instantaneous utilization is between $0.9$ and $1.1$). Such an improper clock rate assignment may result in a significant increase on the energy consumption if the energy consumption of the other tasks is relatively small. This also explains why Algorithms SGA and EGA have very similar performance for the distribution of Type-II since Algorithm EGA only tries to improve the other tasks instead of the large task. When $\epsilon$ is equal to $0.5$, the maximum ratio observed is no more than $1.21$ and the average ratio is no more than $1.10$. When $\epsilon$ is equal to $0.1$, the maximum ratio observed is no more than $1.02$ and the average ratio is no more than $1.01$. In the evaluated cases, setting $\epsilon$ as $0.1$ is the best trade-off on the energy consumption and the running time of Algorithm $\epsilon$-ME.

Figure 2 shows the performance when the workload distribution is Type-III. The average and maximum normalized relaxed energy consumptions for distributions of Type-III are illustrated in Figures 2(a) and 2(b), respectively. Algorithm $\epsilon$-ME still outperforms Algorithms SGA and EGA. The results in Figure 2 show that all of the evaluated algorithms have a smaller normalized relaxed energy consumption with larger number of tasks. This is because that the increased energy consumption resulted from choosing improper clock rates for tasks becomes much less due to the smaller variance of the maximal instantaneous utilization between tasks for larger number of tasks.

## 5. CONCLUSIONS

In this paper, we consider the clock rate assignment problem for task executions on a processor with the capability of dynamic voltage scaling, where there is a finite number of available clock rates. Our objective is to guarantee task deadlines and to minimize the energy consumption at the same time. We propose a polynomial-time $(1 + \epsilon)$-approximation algorithm for the scheduling of periodic real-time tasks, where $\epsilon$ is the tolerable error margin given by users $(1 \geq \epsilon > 0)$. Our solution provides trade-offs among the user's tolerable error and the complexity, including the time complexity and the space complexity. The performance of the proposed algorithm is evaluated by extensive experiments, compared to algorithms proposed in [4]. The proposed algorithm outperforms Algorithms SGA and EGA in [4] for various workloads.

For future research, we shall extend this work for energy-efficient scheduling in multiprocessor environments. We shall also exploit an integrated approach for task scheduling and clock rate assignments along with resource competition.

## REFERENCES

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

[2] 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.

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

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

[5] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-time Systems Journal*, July 1989.

[6] INTEL-XSCALE MICROARCHITECTURE. http://developer.intel.com/design/intelxscale/benchmarks.htm.

[7] 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.