# Energy-aware Scheduling on Multiprocessor Platforms with Devices

Dawei Li, Jie Wu
Dept. of Computer and Information Sciences
Temple Univ., PA
{dawei.li, jiewu}@temple.edu

Keqin Li
Dept. of Computer Science
State Univ. of NY at New Paltz, NY
lik@newpaltz.edu

Kai Hwang
Electrical Eng. and Computer Science
Univ. of South California, CA
kaihwang@usc.edu

***Abstract*—In this paper, we address the problem of energy-aware task scheduling on DVFS-enabled multiprocessors with DPM-enabled device(s). Given a set of frame-based tasks, we aim to derive a scheduling where the device occupation constraint is respected, all of the tasks meet the shared deadline, and the overall system energy consumption, including energy consumed on both processors and devices, is minimized. For the problem when preemption and migration are allowed, after solving the formulated optimization problem, we regard the tasks that require the same device as a single preemptive task. An Execution Time Filling (ETF) process can be applied to derive a scheduling which adopts the optimal frequency setting; then, we propose Algorithm ETFR, which achieves the optimal system energy consumption, and also Reduces the total number of preemptions and migrations. For the problem when tasks are non-preemptive, we regard the tasks that require the same device as a single non-preemptive task. To assign tasks to processors, we adopt the Worst Fit Decreasing (WFD) strategy using tasks' optimal execution times. After task assignment, we readjust the execution frequency of tasks on each processor, such that the system energy consumption of tasks on each processor is minimized. Various analysis, simulations, and experiments verify the strength of our proposed approaches for the two problems.**

***Index Terms*—Dynamic voltage and frequency scaling (DVFS), dynamic power management (DPM), system energy consumption, energy-aware scheduling, execution time filling.**

## I. Introduction

### A. DVFS and DPM

The main design goal of modern computational systems has been to improve computing capability. Recently, the high energy consumption in these systems has become an important issue. To facilitate energy-efficient design, the Dynamic Voltage and Frequency Scaling (DVFS) scheme is widely used [1] [2].

During the past two decades, tremendous works have been done for energy-aware scheduling on DVFS-enabled platforms. We refer the readers to a comprehensive survey [3], which includes energy-aware task scheduling for framed-based tasks, periodic tasks, sporadic tasks, and tasks with precedence constraints. The basic idea of the DVFS strategy is to adjust a processor's processing frequency (online or offline), as long as tasks' deadlines are not violated, to achieve the goal of saving energy. DVFS provides the possibility of minimizing energy consumption, given a performance/timing requirement.

In modern computational systems, various devices or components also consume significant amounts of energy, which is usually non-negligible, compared to the energy consumption on processors. In this paper, we will refer to all units other than processors in the computational systems as "devices." Representative devices include I/O devices, disk drives, displays, memory banks, and network interfaces, etc..

To cope with the energy consumption issues on devices, the Dynamic Power Management (DPM) [4] mechanism is widely employed. For a device with DPM, though the power consumption of a device cannot be adjusted, the device can be put in a low power state when it is idle, to save energy.

In this paper, we address the energy-aware scheduling problem on multiprocessor platforms where processors are DVFS-enabled and devices are DPM-enabled. As an initial study, we assume that a task requires, at most, one device.

### B. Related Work

Various works exist that involve the device energy consumption. They can be roughly classified into two categories, namely static/offline scheduling and dynamic/online scheduling.

[5] discusses energy-aware scheduling on a processor with discrete available speeds and devices. The ideal frequency setting is solved analytically; then, an algorithm is proposed to select the practical frequency from the available values. [6] presents an online I/O device scheduler that takes a predetermined task schedule and device-usage as inputs, and generates a sequence of sleep/working states for each device. Considering the switching overhead of devices, Lu et al. [7] aim to arrange task execution orders so that device idle periods are clustered, which is beneficial for device energy saving. [8] elaborates on executing one application on one DVFS processor, where the multiple devices considered have explicit energy and time transition overheads.

[9] proposes both static scheduling and dynamic scheduling for system energy minimization. [10] considers pure online device scheduling for a platform without the ability of DVFS. It aims to cluster the device idle periods to reduce the device switching overhead. [11] discusses dynamic task scheduling to determine the speed setting for periodic tasks. When device power and processor power is comparable, an algorithm which reduces the system energy is proposed. [12] develops a unified energy management framework for deadline-driven periodic real-time applications with both static and dynamic solution components.

However, all of these works are considering uniprocessors. To the best of our knowledge, up until now, little work has been done on energy-aware scheduling involving both devices and multiprocessor platforms.

## C. *Our Work and Contributions*

In this paper, we deal with the problem of energy-aware task scheduling on DVFS-enabled multiprocessor platforms with DPM-enabled devices. Our main contributions are as follows:

- To the best of our knowledge, we are the first to address energy-aware scheduling on DVFS-enabled multiprocessor platforms, with the consideration of device energy consumption and power management, to minimize the system energy consumption.
- After formulating the basic optimization problem, we derive the scheduling that adopts the optimal frequency setting, and achieves the minimal energy consumption when tasks are preemptive and migrations are allowed.
- When tasks are non-preemptive, by adopting a widely used WF strategy to allocate tasks, we can derive a scheduling that achieves satisfiable energy consumption on average, and whose energy consumption is no greater than $(1+\beta)^{\alpha-1}$ times that of the optimal scheduling, where $\beta \in (0,1)$ is a parameter that can be easily achieved from the optimization solution, and $\alpha \geq 2$ indicates the relationship between a processor's execution frequency and power consumption: $p(f) = f^\alpha + p_0$.

## D. *Paper Organization*

Section II provides the detailed platform model and task model, as well as the problem definition. We address the problem where preemptions and migrations are allowed in Section III. Section IV addresses the problem when tasks are non-preemptive. Experiments and simulations are provided in Section V. Section VI concludes our work and sketches several future directions.

## II. System Model and Problem Definitions

## A. *Platform Model*

We consider a multiprocessor platform with $m$ homogeneous DVFS-enabled processors with frequency range $[0, +\infty)$. Processors can operate in two modes when it is on: *active* mode and *idle* mode. Active mode refers to the state when it is executing some task and the power consumption is the sum of dynamic power, $\gamma f^\alpha, (\gamma > 0, \alpha \geq 2)$, and static power, $p_0^{ac}$. Idle mode refers to the state when it is not executing any task and the power consumption consists of only static power, $p_0^{id}, (p_0^{id} \leq p_0^{ac})$.

The multiprocessor platform has $v$ DPM-enabled devices, denoted by $DvL = \{D_1, \cdots, D_v\}$, where $D_j (j = 1, \cdots, v)$ is the $j$th device. We assume that a device can also be in two modes when it is on: *active* mode when the device is occupied by some task, and *idle* mode when it is not occupied by any task. The power consumption of device $D_j$ in active mode is a constant $p_j^{ac}$; in idle mode, the device immediately enters a low power state $p_j^{id}, (p_j^{id} \leq p_j^{ac})$. Power consumption of an off processor and device is zero.

Though we consider a set of frame-based tasks, these tasks may also have a periodic feature. We assume that switching a processor or device on and off will incur a significant amount of overhead, in terms of both time and energy. Thus, we do not consider switching the processor/device off; while a processor/device can switch between active and idle modes with negligible overhead. We normalize the power consumption of a processor as: $p(f) = f^\alpha + p_0$ when it is executing a task at frequency $f$, and zero when it is idle $(p_0 = (p_0^{ac} - p_0^{id})/\gamma)$. Similarly, a device power is normalized as $p_j = (p_j^{ac} - p_j^{id})/\gamma$ when it is active, and zero when it is idle. From now on, we will use this normalized energy model for processors and devices.

## B. *Task Model*

We consider a set of frame-based tasks, $T = \{\tau_1, \cdots, \tau_n\}$, which are released at time 0, and share a same deadline $D$. Each task may be preemptive or non-preemptive, but they cannot be executed on more than one processor concurrently. $\tau_i$ has an execution requirement $c_i$. Besides, a task may require some device when executing. Denote $DvL_i$ as the list of devices that are required by $\tau_i$. We further assume that a task requires at most one device; then, $DvL_i$ is a one-element subset of $DvL$. The device in $DvL_i$ is requested by $\tau_i$ when, and only when, $\tau_i$ is executing. If $\tau_i$ does not require any device during its execution, then $DvL_i$ is an empty set $\emptyset$. It is necessary to mention that, due to the convexity of the energy consumption function of processors, different parts of a single task must execute at the same frequency in order to achieve less energy consumption. When task $\tau_i$ is executed at frequency $f$, the time to finish its execution is $c_i/f$. The energy consumption on the processor is: $E_{\tau_i}^{Pr}(f) = c_i(f^{\alpha-1} + p_0/f)$; if $DvL_i$ are required, the energy consumption on the device(s) is: $E_{\tau_i}^{Dv}(f) = (\sum_{D_j \in DvL_i} p_j)c_i/f$. [1] Thus, the system energy consumption to execute task $\tau_i$ at frequency $f$ is:

$$\begin{aligned} E_{\tau_i}(f) &= E_{\tau_i}^{Pr}(f) + E_{\tau_i}^{Dv}(f) \\ &= c_i f^{\alpha-1} + (p_0 + (\sum_{D_j \in DvL_i} p_j))c_i/f. \end{aligned}$$

We assume that one device can be occupied by at most one task at any specific time. Thus, if two tasks require the same device, they cannot execute simultaneously, even though there exist idle processors. We are aware that, in practice, some devices can be occupied by more than one task; however, in this case, there should be some upper bound on the number of tasks that can occupy this device at the same time. Considering this aspect, the problem will become more complicated. We will investigate this aspect in our future work; in this initial work, we will stick to the assumption that one device can be occupied by, at most, one task at any specific time. We can also regard that the devices we consider here are some critical resources that cannot be occupied by more than one task.

## C. *Problem Definition*

Given a set of frame-based tasks $\{\tau_1, \tau_2, \cdots, \tau_n\}$ with specified execution requirements and device requirements, our goal is to schedule these tasks on the platform with $m$

[1] In practice, a task's device energy consumption may be independent of its execution time. In this case, only device's occupation constraint should be considered; device's energy consumption need not to be addressed. We do not include this case in our problem formulation, because it is only a simpler case of our problem.

homogeneous processors and $v$ devices, such that all of the tasks meet the deadline $D$, no device is occupied by more than one task at any time, and the system energy consumption is minimized. We denote it as the problem of Energy minimization on Multiprocessor platforms with Devices (EMD). When tasks are Preemptive and migrations are allowed, the problem is denoted as EMDP. When tasks are Non-preemptive and migration is not allowed, the problem is denoted as EMDN.

## III. EMDP

### A. Problem Reformulation

As has been mentioned, we assume that each task should require, at most, one device. Thus, we can classify all of the tasks into $(v+1)$ categories: the first category consists of tasks that do not require any device; the second category consists of tasks that require device $D_1$; the third category consists of tasks that require device $D_2$; $\cdots$; the $(v+1)$th category consists of tasks that require device $D_v$. We denote these categories by $T_0, T_1, T_2, \cdots, T_v$, respectively. Relabel the tasks in these sets such that $T_0 = \{\tau_{0,1}, \cdots, \tau_{0,|T_0|}\}$, $T_1 = \{\tau_{1,1}, \cdots, \tau_{1,|T_1|}\}$, $\cdots$, $T_v = \{\tau_{v,1}, \cdots, \tau_{v,|T_v|}\}$, where $|T_j|$ $(j = 0, 1, \cdots, v)$ is the number of tasks in $T_j$. Also, let $c_{j,i}$ $(i = 1, 2, \cdots, |T_j|)$ be the execution requirement of task $\tau_{j,i}$. Assume that task $\tau_{j,i}$ is executed at frequency $f_{j,i}$, then, the energy consumption to execute $\tau_{j,i}$ can be calculated as $E_{\tau_{j,i}}(f_{j,i})$, where $E_{\tau_{0,i}}(f_{0,i}) = c_{0,i}(f_{0,i}^{\alpha-1} + p_0/f_{0,i})$ and $E_{\tau_{j,i}}(f_{j,i}) = c_{j,i}(f_{j,i}^{\alpha-1} + (p_0+p_j)/f_{j,i}), j = 1, 2, \cdots, v$. The total system energy consumption can be calculated as $\sum_{j=0}^{v}\sum_{i=1}^{|T_j|} E_{\tau_{j,i}}(f_{j,i})$.

Obviously, the frequency setting should at least satisfy the following constraints. First, the execution time of each task that does not require the device is less than or equal to the deadline $D$, i.e., $c_{0,i}/f_{0,i} \le D, \forall i = 1, 2, \cdots, |T_0|$. Second, the occupation time of each device is less than or equal to the deadline $D$, because the device cannot be occupied by more than one task at the same time, i.e., $\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} \le D, \forall j = 1, 2, \cdots, v$. Third, the overall execution time of all of the tasks should be less than or equal to the total available execution time $mD$, i.e., $\sum_{j=0}^{v}\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} \le mD$. Considering these three requirements, the frequency setting problem can be formulated as the following optimization problem:

$$\min \quad \sum_{j=0}^{v}\sum_{i=1}^{|T_j|} E_{\tau_{j,i}}(f_{j,i}) \tag{1}$$
$$s.t. \quad c_{0,i}/f_{0,i} \le D, \forall i = 1, 2, \cdots, |T_0| \tag{2}$$
$$\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} \le D, j = 1, 2, \cdots, v \tag{3}$$
$$\sum_{j=0}^{v}\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} \le mD. \tag{4}$$

which will be referred to as *Prob 1* throughout this paper.

Although device power is considered, the objective function of *Prob 1* is still convex, and this optimization problem is still a convex optimization problem, which can be solved in polynomial time [13]. In our work, we apply the widely used Interior Point method to solve this problem numerically. The complexity of this method is polynomial in terms of the number of variables and the solution accuracy of the optimization problem. This aspect has also been explained in the book [13].

The solution of the problem satisfies some important characteristics, which can be used to solve some simpler cases of the problem, and simplify the problem itself. Denote $\mu_{0,1}$, $\mu_{0,2}$, $\cdots$, $\mu_{0,|T_0|}$ as the KKT multipliers [14] associated with the $|T_0|$ inequalities in (2), $\mu_1, \mu_2, \cdots, \mu_v$, the KKT multipliers associated with the $v$ inequalities (3), and $\mu$, the KKT multiplier associated with the inequality (4). All of these multipliers are nonnegative. Applying the KKT conditions, we have:

$$(\alpha-1)f_{0,i}^{\alpha-2} - \frac{p_0 + \mu_{0,i} + \mu}{f_{0,i}^2} = 0, \forall i = 1, 2, \cdots, |T_0| \tag{5}$$
$$(\alpha-1)f_{j,i}^{\alpha-2} - (p_0 + p_j + \mu_j + \mu)/f_{j,i}^2 = 0 \tag{6}$$
$$\forall j = 1, 2, \cdots, v, \forall i = 1, 2, \cdots, |T_j|$$
$$\mu_{0,i}(c_{0,i}/f_{0,i} - D) = 0, \forall i = 1, 2, \cdots, |T_0| \tag{7}$$
$$\mu_j(\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} - D) = 0, \forall j = 1, 2, \cdots, v \tag{8}$$
$$\mu(\sum_{j=0}^{v}\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} - mD) = 0. \tag{9}$$

Without any confusion, we also denote the optimal frequency setting for this problem as $f_{j,i}$ . There are two important characteristics that should be noticed. First, according to (6), we have $f_{j,i} = \sqrt[\alpha]{(p_0 + p_j + \mu_j + \mu)/(\alpha-1)}, j = 1, 2, \cdots, v$; since $f_{j,i}$ does not depend on $i$, the optimal execution frequency of all of the tasks that require the device $D_j$ should be equal. Second, all of the tasks in $T_j$ require the same device $D_j$, and thus cannot execute simultaneously. Based on these two aspects, we can regard $T_j$ as one single preemptive task. From now on, we will refer $T_j$ to a single task without any confusion, and denote $f_j = f_{j,1} = \cdots = f_{j,|T_j|}$ as the optimal frequency setting for this task. Let $C_j = \sum_{i=1}^{|T_j|} c_{j,i}$ be the execution requirement of task $T_j$. The energy consumption of $T_j$ can be calculated as $E_{T_j}(f_j) = C_j(f_j^{\alpha-1} + (p_0+p_j)/f_j)$. Thus, the optimization problem can be simplified as:

$$\min \quad \sum_{i=0}^{|T_0|} E_{\tau_{0,i}}(f_{0,i}) + \sum_{j=1}^{v} E_{T_j}(f_j) \tag{10}$$
$$s.t. \quad c_{0,i}/f_{0,i} \le D, \forall i = 1, 2, \cdots, |T_0| \tag{11}$$
$$C_j/f_j \le D, j = 1, 2, \cdots, v \tag{12}$$
$$\sum_{i=1}^{|T_0|} c_{0,i}/f_{0,i} + \sum_{j=1}^{v} C_j/f_j \le mD. \tag{13}$$

The above problem is considered to be a simplification of *Prob 1* because the number of variables that need to be determined is significantly reduced, namely, from $(|T_0| + \sum_1^v |T_v|)$ to $(|T_0| + v)$. Recall that the complexity of the Interior Point method, which we have used, is polynomial in terms of the number of variables. Thus, the simplified problem will be solved much more efficiently. We denote the simplified problem as *Prob 2* from now on.

Notice that, although the optimal frequency setting can be achieved by the Interior Point method, it does not guarantee that the practical scheduling which adopts this frequency setting is achievable. In the following, we will construct a schedule that adopts the optimal frequency setting.

**Algorithm 1** ETF($T^u, M_s$)

```
Input: Unscheduled task set $T^u$ and the first available processor $M_s$; the execution requirement, $c_\tau$ for each $\tau \in T^u$, also its optimal frequency setting $f_\tau$, and consequently, the optimal execution time $t_\tau = c_\tau / f_\tau$;
Output: A feasible preemptive schedule that adopts the optimal frequency setting and achieves the minimal energy consumption;
1:  Number of processors that should be used: $N = \lceil \sum_{\tau \in T^u} t_\tau / D \rceil$;
2:  Initialize the execution time on processor $M_j$ as $P_j$, where $P_j = 0$, $\forall j = 1, 2, \cdots, N$;
3:  $j = s$;
4:  while $T^u \neq \emptyset$ do
5:      Pick $\tau \in T^u$;
6:      if $P_j + t_\tau > D$ then
7:          Schedule the first part of $\tau$ on processor $M_{j+1}$ from time 0 to time $P_j + t_\tau - D$; $P_{j+1} = P_j + t_\tau - D$; Schedule the second part of $\tau$ on processor $M_j$ from time $P_j$ to time $D$; $P_j = D$; $j = j + 1$;
8:      else
9:          Schedule $\tau$ on processor $M_j$ from time $P_j$ to time $P_j + t_\tau$; $P_j = P_j + t_\tau$;
10:     $T^u = T^u \setminus \{\tau\}$;
```

### B. A Simple Approach

The Execution Time Filling (ETF) procedure, shown by Algorithm 1, provides a direct way of scheduling: fill in the processors one by one, from the first empty processor, with the execution times of unscheduled tasks $\tau_{0,1}, \tau_{0,2}, \cdots, \tau_{0,|T_0|}$, and tasks $T_1, T_2, \cdots, T_v$ (all of these ($|T_0|+v$) tasks can be in any order). Algorithm 1 provides the details of this scheduling procedure. ETF($\{\tau_{0,1}, \tau_{0,2}, \cdots, \tau_{0,|T_0|}, T_1, T_2, \cdots, T_v\}, M_1$) gives the actual scheduling for a given task set.

In this algorithm, it can be noticed that the number of preemptions and migrations are equal to the number of used processors minus one, i.e., $(N-1)$, in general. By observing some key properties of the optimization problem, we can design another algorithm which reduces the number of preemptions and migrations.

### C. An Improved Approach

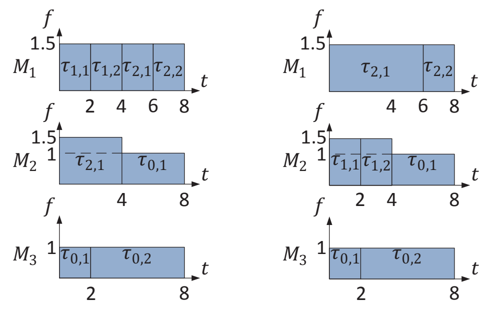
Take a look at (7) and (8). If $\mu_{0,i} \neq 0$, then $c_{0,i}/f_{0,i} = D$; similarly, if $\mu_j \neq 0$, then $\sum_{i=1}^{|T_j|} c_{j,i}/f_{j,i} = D$. We can see in the optimal solution, the execution time of some task(s) (including $T_1, T_2, \cdots, T_v$) might be exactly $D$. If we schedule each such a task on a separate processor, there will not be any preemption and migration on this processor. Thus, the total number of preemptions and migrations will be reduced. Algorithm 2, denoted by ETFR (short for Execution Time Filling with Reduced number of preemptions and migrations), reduces the total number of preemptions and migrations.

### D. Example

Consider a task set consisting of six tasks, whose execution requirements are $c_1 = c_2 = 3$, $c_3 = 9$, $c_4 = 3$, $c_5 = c_6 = 6$ and have a common deadline, $D = 8$. The platform consists

**Algorithm 2** ETFR

```
Input: Optimal solution for *Prob 2*;
Output: A feasible preemptive schedule that adopts the optimal frequency setting, achieves the minimal energy consumption, and reduces the number of preemptions and migrations;
1:  Find $T^r \subset \{\tau_{0,1}, \cdots, \tau_{0,|T_0|}, T_1, \cdots, T_v\}$, where, for each $\tau \in T^r$, $t_\tau = D$;
2:  $T^u = \{\tau_{0,1}, \cdots, \tau_{0,|T_0|}, T_1, \cdots, T_v\} \setminus T^r$;
3:  Schedule the $|T^r|$ tasks in $T^r$ to $|T^r|$ processors separately, from time 0 to $D$; $s = |T^r| + 1$;
4:  ETF($T^u, M_s$);
```



(a) ETF (b) ETFR

Fig. 1. Example for the EMDP problem with multiple devices

of three processors $M_1, M_2, M_3$ and two devices $D_1, D_2$. We assume $\alpha = 3$ and $p_0 = 0$ in this example. The constant powers of $D_1$ and $D_2$ are $p_1 = 19/4$ and $p_2 = 1$, respectively. Tasks' device requirements are: $DvL_1 = DvL_2 = \{D_1\}$, $DvL_3 = DvL_4 = \{D_2\}$, and $DvL_5 = DvL_6 = \emptyset$. According to our approach, we relabel the tasks as $\tau_{1,1}, \tau_{1,2}$, $\tau_{2,1}, \tau_{2,2}$, $\tau_{0,1}, \tau_{0,2}$. Then, $c_{1,1} = c_{1,2} = 3$, $c_{2,1} = 9$, $c_{2,2} = 3$, $c_{0,1} = c_{0,2} = 6$. Solving *Prob 2*, we can get $f_{1,1} = f_{1,2} = f_{2,1} = f_{2,2} = 3/2$, $f_{0,1} = f_{0,2} = 1$. Considering the tasks in the listed order, algorithm ETF gives the schedule in Fig. 1(a), where $\tau_{2,1}$ and $\tau_{0,1}$ are split into two parts. *Two* preemptions and migrations exist in the ETF scheduling. Since the total execution time of $\tau_{2,1}$ and $\tau_{2,2}$ is 8, which is exactly the deadline $D$, the ETFR algorithm first schedules $\tau_{2,1}$ and $\tau_{2,2}$ on a separate processor, as shown in Fig. 1(b). The number of preemption(s) and migration(s) is reduced to *one*.

## IV. EMDN

### A. Analysis

When tasks under consideration are non-preemptive, i.e., one task should be finished on the processor that it is first assigned to without any interruption, the problem (either with one device or multiple devices) involves task set partitioning. The energy-aware task set partitioning problem, on the pure homogeneous multiprocessor platforms without any device, is NP-complete [15]; with the consideration of device occupation constraint and device power, the problem is made more complex. Instead of seeking the optimal scheduling, we will provide a heuristic, with a reasonable approximation ratio and

**Algorithm 3** WFD for the EMDN problem

```
Input: Optimal solution for Prob 2;
Output: A feasible non-preemptive schedule that attempts to achieve the minimal energy consumption;
1: $t^{sum} = \sum_{i=1}^{|T_0|} c_{0,i}/f_{0,i} + \sum_{j=1}^{v} C_j/f_j$; Number of processors used: $N = \lceil t^{sum}/D \rceil$;
2: Initialize $P_k = 0$, for $k = 1, 2, \cdots, N$;
3: Sort tasks in $T^u = \{\tau_{0,1}, \cdots, \tau_{0,|T_0|}, T_1, \cdots, T_v\}$ in the non-increasing order of their optimal execution times;
4: while $T^u \neq \emptyset$ do
5:     $\tau$ = the first task in the sorted $T^u$;
6:     Find $P_{k^*} = \min\{P_k | k = 1, 2, \cdots, N\}$ (ties can be broken arbitrarily);
7:     Allocate $\tau$ to processor $M_{k^*}$;
8:     $P_{k^*} = P_{k^*} + t_\tau$; $T^u = T^u \setminus \{\tau\}$;
9: Readjust the execution frequencies of tasks on processors $M_1$, $M_2$, $\cdots$, $M_N$ such that all tasks meet the deadline and tasks' system energy consumption on each processor is minimized.
```

excellent average performance, based on the optimal solution to *Prob 2*. We will first prove the following Lemma:

*Lemma 1:* The energy consumption of the optimal non-preemptive scheduling for the EMDN problem will be no less than that of the optimal solution of *Prob 2*.

*Proof:* This fact is obvious, since the execution frequency settings in any non-preemptive schedule should also satisfy all the constraints in *Prob 1*, and *Prob 1* and *Prob 2* have the same optimal objective value. ■

### B. Algorithm

In our scheduling, we also regard each task set $T_j (j = 1, \cdots, v)$ as a single non-preemptive task. Our guidance to construct the scheduling is to let more tasks operate on the optimal frequency setting for *Prob 2*; if the optimal frequency setting cannot be guaranteed, we desire the practical frequency setting to be close to the optimal frequency setting.

Thus, we first find each task $\tau \in \{\tau_{0,1}, \cdots, \tau_{0,|T_0|}, T_1, \cdots, T_v\}$, whose execution time $t_\tau = D$; then, pick it out and schedule it on a separate processor. Then, we need to consider scheduling the remaining tasks, whose execution times are less than $D$, on the remaining processors. We adopt the widely used Worst Fit Decreasing (WFD) strategy: first, sort the tasks in non-increasing order of their optimal execution times; then, assign the next task to the processor, which has the least accumulative execution time among all processors that should be used. Algorithm 3 (denoted by WFD for short) describes the details of our scheduling scheme for the EMDN problem.

After allocating the tasks to processors, since all of the processors are independent, tasks on different processors are also independent; we can readjust the frequency setting for all of the tasks such that each task meets the deadline $D$ and tasks' system energy consumption on each processor is minimized, by solving the following problem:

$$\begin{array}{ll} \min & C_{k,0}(F_{k,0}^{\alpha-1} + \frac{p_0}{F_{k,0}}) + \sum_{j=1}^{v} C_{k,j}(F_{k,j}^{\alpha-1} + \frac{p_0+p_j}{F_{k,j}}) \\ s.t. & \sum_{i=0}^{v} C_{k,j}/F_{k,j} \leq D. \end{array}$$

where $C_{k,j}$ is the total execution requirement of category $T_j$ tasks that are allocated to the processor $M_k$ ($j = 0, 1, \cdots, v$, $k = 1, 2, \cdots, N$); $F_{k,j}$ is the optimal frequency setting for the category $T_j$ tasks on processor $M_k$.

Again, the optimal execution time (solved for *Prob 2*) of each task $\tau \in T^u$ can be calculated as $t_\tau = c_\tau/f_\tau$. Let $\beta D$ be the maximal execution time of tasks in $T^u$, but not equal to $D$. Assume that, up to line 8, the execution time on processor $M_j, (j = 1, \cdots, N)$ is $L_j$ and that, the maximal and the minimal values among $L_j$'s are $L_{max}$ and $L_{min}$, respectively. The following Lemma can be easily shown.

*Lemma 2:* $L_{max} \leq (1 + \beta)D$.

*Proof:* First, $L_{min} \leq D$; otherwise, the total execution time of $T^u$ will be greater than $ND$, which is a contradiction. Second, $L_{max} - L_{min} \leq \beta D$, which can also be verified by contradiction. Assume that $L_{max} - L_{min} > \beta D$; let the execution time of the last task of $L_{max}$ be $\beta_1 D$. Since $\beta_1 D \leq \beta D$, $L_{max} - \beta_1 D \geq L_{max} - \beta D > L_{min}$. According to the WFD strategy, the last task of $L_{max}$ will not be assigned to $L_{max}$, which contradicts the fact. Thus, $L_{max} - L_{min} \leq \beta D$. Based on the above two aspects, $L_{max} \leq (1 + \beta)D$. ■

*Theorem 1:* The energy consumption of the scheduling $\mathcal{S}$ derived by Algorithm 3 is no greater than $(1 + \beta)^{\alpha-1}$ times that of the non-preemptive optimal energy consumption.

*Proof:* After the WFD allocation, we can increase each task's execution frequency to $\max\{L_{max}/D, 1\}$ $(\leq 1 + \beta)$ times its original optimal frequency (solved for *Prob 2*). Denote this scheduling by $\mathcal{S}_1$. By doing this, the energy consumption on the devices will not increase, while the energy consumption on processors will be no greater than $(1+\beta)^{\alpha-1}$ times the energy consumption of *Prob 2*. Notice that $\mathcal{S}$ is further optimized based on the same task set partitioning as that of $\mathcal{S}_1$. Obviously, the energy consumption of $\mathcal{S}$ is no greater than that of $\mathcal{S}_1$. Combining *Lemma 1*, we can conclude that the energy consumption of the scheduling $\mathcal{S}$ derived by Algorithm 3 is no greater than $(1 + \beta)^{\alpha-1}$ times of the non-preemptive optimal energy consumption. ■

## V. Experiments and Simulations

### A. Simulation Settings

There are various parameters that might affect the performance of a scheduling algorithm, namely, the number of processors, the number of devices, the constant device power(s), the number of tasks (including the number of tasks that do not require any device, and the number of tasks that require each device), the execution requirements of tasks, the percentage of the requirement of the tasks that require the device(s) and the tasks that do not require device(s), the common deadline of the task set, etc.. We notice that many of these parameters are correlated, and that some of the values should be set by referring to practical situations.

Considering the number of processors and the number of devices, the relative number makes a difference. In our simulation, we set the number of processors to $m = 4$, which is a common configuration of multiprocessor platforms, and only vary the number of devices. As for the tasks' characteristics, we notice that, the execution requirement percentage has the most significant influence. For each task that requires a device, we generate a uniformly distributed random number within $[15, 25]$ as its execution requirements. Another important parameter to be designed is the common deadline of a task set. We assume that, when a user submits a task set, he/she is roughly aware of the average capability of the computational system, and thus, will set a deadline that is practically reasonable. We assume that the normal execution frequency is around 1, which has been normalized by some reference value. For all our simulations, a reasonable deadline can be $D = \max\{\sum_{j=0}^{v}\sum_{i=1}^{|T_j|} c_{j,i}/4, \max\{c_{0,1}, \cdots, c_{0,|T_0|}, C_1, \cdots, C_v\}\}$.

For the two problems with one device, we fix the number of tasks that require the device to be 16, i.e., $|T_1| = 16$; we also generate a random execution requirement within $[15, 25]$ for tasks that do not require the device, and choose the number of tasks that do not require the device, $|T_0|$ to be 8, 16, and 32. Denote these three settings as Setting 1, 2 and 3, respectively. In all these settings, we vary the device power from 0 to 2, with a step size of 0.1, while $p_0$ is set as 0.1 (if not particularly specified). Without loss of generality, we set $\alpha = 3$ in the simulations.

For the two problems with multiple devices, we vary the number of devices, $v$, from 1 to 12 with a step size of 1. Since we treat $T_j$ as a single task, the total requirement of $T_j$ matters. We fix the number of tasks in $T_j$ to be 4. Under this environment, we further conduct two experiment groups. In the first group, for a given $v$, we vary $|T_0|$ as $8, 16$, and $32$; execution requirements of tasks in $T_0$ are also generated within $[15, 25]$, with the mean value being 20. In the second group, for each given $v$, we fix $|T_0| = 4v$, which is equal to the total number of tasks that require a device; additionally, we vary the execution requirement generation range for tasks in $T_0$ such that the percentage $pct$ of the $T_0$ tasks' total requirement is about $20\%, 50\%$, and $80\%$ of the overall execution requirement. Specifically, the generation range is set to be $[\lceil 15pct/(1-pct)\rceil, \lceil 25pct/(1-pct)\rceil]$, with the mean value being $20pct/(1-pct)$. Due to the limited space, we only show the results for one set of device powers, with each device's power around 0.8, randomly generated within $[0.6, 1]$.

For each specific simulation setting, we calculate the normalized energy consumption values, which are normalized by the optimal energy consumption of *Prob 2*.

## B. EMDP

As has been mentioned, our proposed algorithm is itself optimal in terms of minimizing overall system energy consumption. We compare our scheduling with one that only considers processor energy minimization. For a scheduling that only considers the processor energy consumption, it is easy to notice that all of the constraints (2) to (4) should also be satisfied; however, the objective function only consists of energy consumption on processors. The optimal frequency setting that minimizes the processor energy consumption can be achieved. We can also construct the preemptive scheduling that adopts this optimal frequency setting by the Execution Time Filling (ETF) process. We denote this scheduling as Minimizing Pure Processor Energy Scheduling (MPPES). We denote "NEC of MPPES" as the Normalized system Energy Consumption of MPPES, which is normalized by the optimal energy consumption of *Prob 2*. We also denote "NECP of MPPES" as the Normalized Energy Consumption of Processors of MPPES.

Fig. 2(a) shows the NEC and NECP of MPPES for different device power values for the three settings, where NEC and NECP of MPPES 1, 2, 3 represent the values in Settings 1, 2, and 3, respectively. For each setting, when the device power is small, MPPES's system energy consumption is very close to the optimal solution. When the device power increases, the NEC of MPPES also increases significantly, and will be much greater than the optimal system energy consumption. This phenomenon indicates that when the device power is non-negligible, a scheduling that considers both device power and processor power should be applied to minimize system energy consumption. Given a constant device power, as the number of tasks in $T_0$ increases (from Setting 1 to Setting 3), though the percentage of energy consumption on processors is increased, the system's NEC still remains at a relatively high level.

For the EMDP problem with multiple devices, the NEC and NECP of MPPES are provided in Table I and Table II for the two experiment groups, respectively, both in the rows labeled "NEC of MPPES" and "NECP of MPPES." We can see that, as the number of devices increases, the NEC of MPPES tends to increase, and it remains at a high level. This aspect suggests that when device power is non-negligible, especially when device number is significant, our scheduling ETFR, which aims to minimize overall system energy consumption, should be applied.

## C. EMDN

We compare the energy consumption of our proposed scheduling algorithm with several other algorithms as well as some closely related values, to evaluate the performance of our proposed algorithm for the EMDN problem. Denote "NEC of WFD" as the Normalized Energy Consumption (NEC) of our proposed algorithm. We will compare the NEC of our scheduling algorithm with the following algorithms and values.

Algorithm DWFN: After solving *Prob 2*, instead of picking out each task, whose execution time is exactly $D$, to schedule it on a separate processor, this approach Directly applies the WF strategy to all of the tasks in $T$, Not involving sorting the tasks. [2]

[2]We do not apply WFD strategy when allocating tasks, because if WFD strategy is used, the resulting scheduling will be the same as our proposed scheduling. As for this fact, it is not mandatory to pick out the tasks, whose execution times are exactly $D$, if the WFD strategy is used.

TABLE I
GROUP 1 EXPERIMENTS FOR THE PROBLEMS WITH MULTIPLE DEVICES

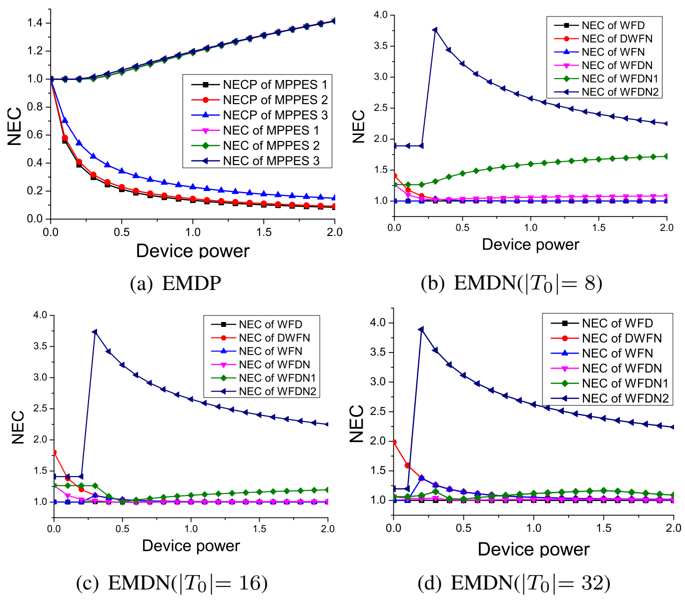
| $\vert T_0\vert$ | NECs | Number of devices, $v$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| | NECP of MPPES | 0.7296 | 0.7466 | 0.7097 | 0.6686 | 0.6620 | 0.6515 | 0.6426 | 0.6397 | 0.6268 | 0.6233 | 0.6246 | 0.6183 |
| 8 | NEC of MPPES | 1.0000 | 1.0089 | 1.0086 | 1.0073 | 1.0070 | 1.0065 | 1.0060 | 1.0058 | 1.0049 | 1.0046 | 1.0047 | 1.0042 |
| | NEC of WFD | 1.0103 | 1.0210 | 1.0008 | 1.0000 | 1.0016 | 1.0008 | 1.0034 | 1.0001 | 1.0021 | 1.0019 | 1.0013 | 1.0001 |
| | NEC of DWFN | 1.4008 | 1.3485 | 1.1210 | 1.0008 | 1.0553 | 1.0767 | 1.0443 | 1.0021 | 1.0255 | 1.0421 | 1.0213 | 1.0027 |
| | NEC of WFN | 1.0164 | 1.3485 | 1.1210 | 1.0008 | 1.0553 | 1.0767 | 1.0443 | 1.0021 | 1.0255 | 1.0421 | 1.0213 | 1.0027 |
| | NEC of WFDN | 1.1232 | 1.1225 | 1.0359 | 1.0064 | 1.0377 | 1.0197 | 1.0582 | 1.0081 | 1.0233 | 1.0259 | 1.0229 | 1.0054 |
| | NEC of WFDN1 | 1.2885 | 1.2215 | 1.0733 | 1.0151 | 1.0884 | 1.0484 | 1.1435 | 1.0210 | 1.0627 | 1.0704 | 1.0621 | 1.0153 |
| | NEC of WFDN2 | 2.2703 | 3.7022 | 3.2166 | 2.8700 | 2.6773 | 2.4234 | 2.1701 | 2.1061 | 1.9319 | 1.8739 | 1.7713 | 1.7000 |
| | NECP of MPPES | 0.8628 | 0.8145 | 0.7505 | 0.7457 | 0.7252 | 0.7011 | 0.6867 | 0.6772 | 0.6705 | 0.6642 | 0.6543 | 0.6489 |
| 16 | NEC of MPPES | 1.0067 | 1.0082 | 1.0089 | 1.0089 | 1.0088 | 1.0084 | 1.0080 | 1.0077 | 1.0074 | 1.0071 | 1.0066 | 1.0063 |
| | NEC of WFD | 1.0001 | 1.0075 | 1.0036 | 1.0005 | 1.0022 | 1.0003 | 1.0010 | 1.0000 | 1.0011 | 1.0003 | 1.0009 | 1.0000 |
| | NEC of DWFN | 1.3359 | 1.1619 | 1.0863 | 1.0016 | 1.0534 | 1.0408 | 1.0244 | 1.0017 | 1.0278 | 1.0251 | 1.0120 | 1.0010 |
| | NEC of WFN | 1.3359 | 1.1619 | 1.0863 | 1.0016 | 1.0534 | 1.0408 | 1.0244 | 1.0017 | 1.0278 | 1.0251 | 1.0120 | 1.0010 |
| | NEC of WFDN | 1.0133 | 1.0962 | 1.0490 | 1.0254 | 1.0304 | 1.0185 | 1.0217 | 1.0028 | 1.0154 | 1.0123 | 1.0278 | 1.0013 |
| | NEC of WFDN1 | 1.0193 | 1.1515 | 1.0896 | 1.0474 | 1.0596 | 1.0391 | 1.0476 | 1.0064 | 1.0358 | 1.0291 | 1.0673 | 1.0032 |
| | NEC of WFDN2 | 3.3682 | 2.6639 | 2.6764 | 2.2774 | 2.0291 | 2.0726 | 1.9676 | 1.8573 | 1.7553 | 1.7088 | 1.6869 | 1.6281 |
| | NECP of MPPES | 0.9241 | 0.8827 | 0.8405 | 0.8090 | 0.7816 | 0.7664 | 0.7479 | 0.7352 | 0.7289 | 0.7180 | 0.7052 | 0.6964 |
| 32 | NEC of MPPES | 1.0041 | 1.0060 | 1.0075 | 1.0083 | 1.0087 | 1.0089 | 1.0089 | 1.0089 | 1.0088 | 1.0087 | 1.0085 | 1.0083 |
| | NEC of WFD | 1.0032 | 1.0027 | 1.0018 | 1.0000 | 1.0008 | 1.0006 | 1.0009 | 1.0000 | 1.0007 | 1.0004 | 1.0006 | 1.0000 |
| | NEC of DWFN | 1.0887 | 1.0461 | 1.0266 | 1.0009 | 1.0210 | 1.0294 | 1.0191 | 1.0024 | 1.0188 | 1.0131 | 1.0085 | 1.0010 |
| | NEC of WFN | 1.0887 | 1.0461 | 1.0266 | 1.0009 | 1.0210 | 1.0294 | 1.0191 | 1.0024 | 1.0188 | 1.0131 | 1.0085 | 1.0010 |
| | NEC of WFDN | 1.0311 | 1.0625 | 1.0627 | 1.0013 | 1.0336 | 1.0192 | 1.0405 | 1.0021 | 1.0206 | 1.0158 | 1.0259 | 1.0009 |
| | NEC of WFDN1 | 1.0405 | 1.0871 | 1.0945 | 1.0022 | 1.0575 | 1.0341 | 1.0748 | 1.0041 | 1.0402 | 1.0319 | 1.0537 | 1.0020 |
| | NEC of WFDN2 | 2.0442 | 1.8740 | 1.8650 | 1.7624 | 1.7606 | 1.6693 | 1.6143 | 1.5856 | 1.5732 | 1.5069 | 1.5035 | 1.4905 |

TABLE II
GROUP 2 EXPERIMENTS FOR THE PROBLEMS WITH MULTIPLE DEVICES

| $T_0\%$ | NECs | Number of devices, $v$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| | NECP of MPPES | 0.6302 | 0.6455 | 0.7039 | 0.6957 | 0.7063 | 0.7056 | 0.7033 | 0.7030 | 0.7012 | 0.6998 | 0.7018 | 0.7125 |
| 20% | NEC of MPPES | 1.0000 | 1.0000 | 1.0084 | 1.0083 | 1.0085 | 1.0085 | 1.0084 | 1.0084 | 1.0084 | 1.0084 | 1.0084 | 1.0086 |
| | NEC of WFD | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0014 | 1.0004 | 1.0005 | 1.0003 | 1.0001 | 1.0000 | 1.0000 | 1.0000 |
| | NEC of DWFN | 1.1663 | 1.3096 | 1.1237 | 1.0003 | 1.0768 | 1.0459 | 1.0218 | 1.0036 | 1.0117 | 1.0159 | 1.0108 | 1.0002 |
| | NEC of WFN | 1.0000 | 1.0029 | 1.1237 | 1.0003 | 1.0768 | 1.0459 | 1.0218 | 1.0036 | 1.0117 | 1.0159 | 1.0108 | 1.0002 |
| | NEC of WFDN | 1.0000 | 1.0065 | 1.0066 | 1.0056 | 1.0156 | 1.0140 | 1.0179 | 1.0081 | 1.0043 | 1.0008 | 1.0024 | 1.0025 |
| | NEC of WFDN1 | 1.0000 | 1.0191 | 1.0138 | 1.0121 | 1.0325 | 1.0292 | 1.0376 | 1.0171 | 1.0091 | 1.0016 | 1.0052 | 1.0052 |
| | NEC of WFDN2 | 2.1875 | 1.7525 | 3.1139 | 2.6022 | 2.1872 | 1.9656 | 1.8687 | 1.7418 | 1.6196 | 1.5577 | 1.4946 | 1.4690 |
| | NECP of MPPES | 0.6325 | 0.6013 | 0.7037 | 0.7049 | 0.7052 | 0.6984 | 0.7068 | 0.7008 | 0.7019 | 0.7022 | 0.7001 | 0.7014 |
| 50% | NEC of MPPES | 1.0000 | 1.0014 | 1.0084 | 1.0085 | 1.0085 | 1.0083 | 1.0085 | 1.0084 | 1.0084 | 1.0084 | 1.0084 | 1.0084 |
| | NEC of WFD | 1.0000 | 1.0014 | 1.0031 | 1.0004 | 1.0009 | 1.0000 | 1.0001 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| | NEC of DWFN | 1.1650 | 1.2741 | 1.1313 | 1.0051 | 1.0507 | 1.0496 | 1.0242 | 1.0006 | 1.0186 | 1.0157 | 1.0111 | 1.0001 |
| | NEC of WFN | 1.0002 | 1.1283 | 1.1313 | 1.0051 | 1.0507 | 1.0496 | 1.0242 | 1.0006 | 1.0186 | 1.0157 | 1.0111 | 1.0001 |
| | NEC of WFDN | 1.0000 | 1.0141 | 1.0634 | 1.0237 | 1.0240 | 1.0043 | 1.0121 | 1.0014 | 1.0017 | 1.0046 | 1.0049 | 1.0002 |
| | NEC of WFDN1 | 1.0000 | 1.0513 | 1.1299 | 1.0492 | 1.0500 | 1.0092 | 1.0252 | 1.0029 | 1.0035 | 1.0097 | 1.0106 | 1.0005 |
| | NEC of WFDN2 | 2.5703 | 3.7999 | 3.3033 | 2.5675 | 2.2189 | 2.0217 | 1.8126 | 1.7040 | 1.6023 | 1.5713 | 1.4972 | 1.4913 |
| | NECP of MPPES | 0.6377 | 0.6374 | 0.6932 | 0.7048 | 0.7075 | 0.6990 | 0.7048 | 0.7034 | 0.6995 | 0.7008 | 0.6981 | 0.7031 |
| 80% | NEC of MPPES | 1.0000 | 1.0000 | 1.0082 | 1.0085 | 1.0085 | 1.0083 | 1.0085 | 1.0084 | 1.0083 | 1.0084 | 1.0083 | 1.0084 |
| | NEC of WFD | 1.0000 | 1.0000 | 1.0041 | 1.0019 | 1.0001 | 1.0003 | 1.0001 | 1.0000 | 1.0001 | 1.0003 | 1.0000 | 1.0000 |
| | NEC of DWFN | 1.1642 | 1.3519 | 1.1481 | 1.0020 | 1.0703 | 1.0566 | 1.0321 | 1.0013 | 1.0116 | 1.0132 | 1.0133 | 1.0007 |
| | NEC of WFN | 1.0000 | 1.0000 | 1.1481 | 1.0020 | 1.0703 | 1.0566 | 1.0321 | 1.0013 | 1.0116 | 1.0132 | 1.0133 | 1.0007 |
| | NEC of WFDN | 1.0000 | 1.0000 | 1.0471 | 1.0269 | 1.0084 | 1.0208 | 1.0090 | 1.0022 | 1.0038 | 1.0134 | 1.0004 | 1.0010 |
| | NEC of WFDN1 | 1.0000 | 1.0000 | 1.1002 | 1.0559 | 1.0175 | 1.0442 | 1.0188 | 1.0046 | 1.0082 | 1.0284 | 1.0008 | 1.0020 |
| | NEC of WFDN2 | 2.5332 | 1.6993 | 3.3242 | 2.5612 | 2.1611 | 2.0626 | 1.8138 | 1.6923 | 1.6637 | 1.5900 | 1.5176 | 1.4727 |

Algorithm WFN: After solving *Prob 2*, this approach also picks out the tasks whose optimal execution time is exactly $D$, to assign it to a separate processor; after that, however, it applies the WF strategy, also Not involving sorting the tasks.

Algorithm WFDN: After allocating the tasks in the same way as that of our proposed algorithm, to ensure schedulability, this algorithm adopts the simplest method, which is to increase the execution frequency of each task to $\max\{L_{max}/D, 1\}$ times that of its optimal frequency setting (solved for *Prob 2*). The normalized energy consumption of this algorithm can be calculated; we denote it by "NEC of WFDN," short for WFD strategy Not involving optimally readjusting the frequency setting. The processor energy consumption will be $(\max\{L_{max}/D, 1\})^2$ times that of the processor energy consumption in the optimal solution; while the device energy consumption will not increase. Thus, the overall energy consumption of Algorithm WFDN will be less than $(\max\{L_{max}/D, 1\})^2$ times that of the preemptive optimal solution. We have also proven that $(L_{max}/D)^2 < (1 + \beta)^2$. We include these values in our comparisons, and denote $(\max\{L_{max}/D, 1\})^2$ and $(1 + \beta)^2$ by "NEC of WFDN1" and "NEC of WFDN2," respectively.



(a) EMDP (b) EMDN($|T_0|= 8$) (c) EMDN($|T_0|= 16$) (d) EMDN($|T_0|= 32$)

Fig. 2. Simulation Results with One Device

Fig. 2(b), Fig. 2(c), and Fig. 2(d) present the results for the problem with one device in the settings 1, 2, and 3, respectively. We notice that when device static power $p_0$ reduces the difference of the comparing values increases. To show the results more clearly, we choose $p_0 = 0$ for simulations in Fig. 2(b), Fig. 2(c), and Fig. 2(d). When device power is very small, the optimal frequency setting for *Prob 2* also attempts to stretch the tasks as much as possible. Thus, $T_1$ has a great chance to have an execution time of $D$. Assigning $T_1$ to a separate processor (as in WFD and WFN) has great advantages over Algorithm DWF. When the device power is significant, $T_1$ will not have an execution time of $D$; in this case, Algorithm WFN and Algorithm DWF are actually the same.

The results for various numbers of devices are also provided in Table I and Table II. In Table I, the first column, $|T_0|$, means the number of tasks in $T_0$. In Table II, the first column, $T_0\%$, means the approximate execution requirement percentage of the tasks in $T_0$. Related values are listed from the third row to the eighth row for each simulation setting. We can see that our proposed WFD strategy always achieves the lowest energy consumption, and near optimal solution. Also, it is always upper bounded by the the two values: "NEC of WFDN1" and "NEC of WFDN2."

## VI. Conclusion and Future Work

This paper addresses the problem of minimizing system energy consumption on multiprocessor platforms with devices, given a set of frame-based tasks, some of which require device(s) and some of which do not. We consider the problem both when preemption and migration are allowed (EMDP problem), and when preemption and migration are not allowed (EMDN problem). For the EMDP problem, we formulated the problem as a convex optimization problem, and numerically solve it by applying the widely used Interior Point method. Based on the optimal solution, we construct a schedule that adopts the optimal frequency setting and achieves the optimal system energy consumption. For the EMDN problem, we derive a schedule that achieves near-optimal energy consumption on average and has an approximation ratio of $(1 + \beta)^{\alpha-1} (< 2^{\alpha-1})$, where $\beta$ is within $(0, 1)$, and can be easily achieved from the optimal solution of the originally formulated convex optimization problem and $\alpha \geq 2$ indicates the relationship between a processor's power consumption and its execution frequency.

Our future work will investigate the problem when one device can be required by more than one task, and/or each task can require more than one device. Further, device switching overhead (both time and energy) can be included in the problems.

## References

[1] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, Oct. 1994, pp. 8 – 11.

[2] A. Chandrakasan, A. Burstein, and R. W. Brodersen, "A low power chipset for portable multimedia applications," in *Solid-State Circuits Conference, 1994. Digest of Technical Papers., IEEE International*, Feb. 1994, pp. 82 – 83.

[3] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug. 2007, pp. 28 – 38.

[4] H. El Ghor, M. Chetto, R. H. Chehade, and G. Nachouki, "Towards multiprocessor scheduling in distributed real-time systems under energy constraints," in *Proceedings of the International Conference on Advances in Computational Tools for Engineering Applications*, July 2009, pp. 355 – 361.

[5] R. Jejurikar and R. Gupta, "Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems," in *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, Aug. 2004, pp. 78 – 81.

[6] V. Swaminathan and K. Chakrabarty, "Energy-conscious, deterministic i/o device scheduling in hard real-time systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 7, pp. 847 – 858, July 2003.

[7] Y.-H. Lu, L. Benini, and G. De Micheli, "Low-power task scheduling for multiple devices," in *Proceedings of the Eighth International Workshop on Hardware/Software Codesign*, May 2000, pp. 39 – 43.

[8] V. Devadas and H. Aydin, "On the interplay of dynamic voltage scaling and dynamic power management in real-time embedded applications," in *Proceedings of the 8th ACM international conference on Embedded software*, 2008.

[9] C.-Y. Yang, J.-J. Chen, C.-M. Hung, and T.-W. Kuo, "System-level energy-efficiency for real-time tasks," in *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, May 2007, pp. 266 – 273.

[10] H. Cheng and S. Goddard, "Online energy-aware i/o device scheduling for hard real-time systems," in *Proceedings of Design, Automation and Test in Europe*, vol. 1, Mar. 2006, p. 6 pp.

[11] J. Zhuo and C. Chakrabarti, "Energy-efficient dynamic task scheduling algorithms for dvs systems," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 17:1 – 17:25, 2008.

[12] V. Devadas and H. Aydin, "Dfr-edf: A unified energy management framework for real-time systems," in *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 121 – 130.

[13] S. Boyd and L. Vandenberghe, "Convex optimization," in *Cambridge University Press*, 2004.

[14] E. Chong and S. Zak, *An Introduction to Optimization*, ser. Wiley Series in Discrete Mathematics and Optimization. John Wiley & Sons, 2008.

[15] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 1484 – 1497, Nov. 2008.