Flow Time Minimization under Energy Constraints*

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

Kazuo Iwama
School of Informatics
Kyoto University
Yoshida-Honmachi
Kyoto, Japan
iwama@kuis.kyoto-u.ac.jp

Tei-Wei Kuo and Hseuh-I Lu
Department of Computer Science and Information Engineering
National Taiwan University
Taiwan
{ktw, hil}@csie.ntu.edu.tw

ABSTRACT

Power-aware and energy-efficient designs play important roles for modern hardware and software designs, especially for embedded systems. This paper targets a scheduling problem on a processor with the capability of dynamic voltage scaling (DVS), which could reduce the power consumption by slowing down the processor speed. The objective of the targeting problem is to minimize the average flow time of a set of jobs under a given energy constraint, where the flow time of a job is defined as the interval length between the arrival and the completion of the job. We consider two types of processors, which have a continuous spectrum of the available speeds or have only a finite number of discrete speeds. Two algorithms are given: (1) An algorithm is proposed to derive optimal solutions for processors with a continuous spectrum of the available speeds. (2) A greedy algorithm is designed for the derivation of optimal solutions for processors with a finite number of discrete speeds. The proposed algorithms are extended to cope with jobs with different weights for the minimization of the average weighted flow time. The proposed algorithms are also evaluated with comparisons to schedules which execute jobs at a common effective speed.

Keywords

Energy-aware systems, Scheduling, Flow time minimization, Dynamic voltage scaling.

1. INTRODUCTION

With the advanced technology of circuit designs, many modern processors, such as Intel XScale, can operate at different processor speeds dynamically. Dynamic voltage scaling (DVS) has been adopted in many systems to reduce the supply voltage and the execution speed dynamically. Reducing the processor speed can certainly reduce the power consumption but may lead to the violation of the timing requirements of the systems. To prolong the lifetime of battery-powered embedded systems under the satisfaction of performance, it is desirable to reduce the energy consumption as much as possible under the timing constraints. The energy-efficient scheduling problem is to compromise between the energy consumption and performance degradation.

Since embedded devices generally have real-time requirements to guarantee the stability of the provided services, such as video decoding or wireless communication, the minimization of the energy consumption for real-time systems with the DVS capability has been an important issue in the past decade. Researchers have proposed various scheduling algorithms to minimize the energy consumption for periodic hard real-time tasks under different deadline assumptions [3, 5, 7, 9, 11, 12, 13, 15, 16, 26]. When energy-efficient scheduling of aperiodic real-time jobs is considered, energy-efficient scheduling for uniprocessor environments with a continuous speed spectrum was explored in [5, 11, 26]. Scheduling algorithms have been also proposed for the minimization of the energy consumption when there is a finite number of speeds for a processor with negligible speed transition overheads [8, 12, 14].

In addition to the minimization of energy consumption, another critical issue for energy-aware systems is to maximize the system performance under a given energy constraint. For example, consider a system that is powered by a re-chargeable battery equipped with a solar panel. The available energy consumption to execute tasks is limited at night. For such a system, the scheduler should try to maximize the system performance under the energy constraint instead of pursuing the minimization of energy consumption. For example, researchers in [6, 10, 21, 22] targeted on the maximization of the total system reward under given timing and energy constraints, while Alenawy et al. [2] considered the minimization of dynamic faults for soft real-time systems under a given energy constraint.

Previous researches mostly focus on systems with timing constraints. However, not all jobs have natural deadlines associated with them. For general operating systems such as Windows and Linux, schedulers are not deadline-driven. A fundamental performance metric for job scheduling could be on the minimization of the average flow time, as known as the average response time, or on the minimization of the (maximum) completion time. The objective of the minimization of the completion time is pursued for systems that treat the execution of all the jobs as an execution entity, such as defense and control applications. The minimization of the average flow time is pursued in multimedia or on-line transaction processing applications. This metric reveals the time that a job has to wait between its arrival time and its completion time. Distinct from the measurement on the minimization of the completion time, the average flow time treats all jobs as different entities for performance measurement.

The minimization of the completion time under a given energy constraint can be obtained easily for DVS systems by the convexity of the power consumption functions. This paper mainly considers the performance metric on the minimization of the average flow time of a given job set on a DVS processor under a given energy constraint. However, only limited work has been known for such energy-constrained systems. Pruhs et al. [18] and Albers et al. [1] explore the scheduling of jobs with equal computation requirements under different arrival times on a processor with a continuous spectrum of the available speeds. Processors under considerations in this paper might be with a continuous spectrum of the available speeds between the upper-bounded and lower-bounded speeds or with a finite number of available speeds. We show that the minimization of the average flow time for a set of jobs with the same

*Support in parts by research grants from ROC National Science Council NSC-95-2752-E-002-008-PAE, NSC-95-2219-E-002-014, and NSC-95-2221-E-002-077, and from KAKENHI (16092101, 16092215, 16300002).

arrival/ready time could be derived in polynomial time for the processor models considered in this paper. For processors with a continuous spectrum of the available speeds, we present an algorithm based on the well-known Lagrange Multiplier Method [20]. However, most modern processors can only operate at a finite number of discrete available speeds. Examples are the ARM7D processor (20 or 33 MHz) [23], the Crusoe processor by Transmeta (200-700 MHz in 33MHz steps) [25], the Intel StrongARM SA1100 processor (59-221 MHz in 14.7MHz steps) [23], and the Intel XScale (150-1000 MHz with 5 different levels) [24]. Hence, we also develop a greedy algorithm for processors with a finite number of discrete available speeds. Both of the proposed algorithms are proved to derive optimal solutions under different settings on the processor types. We also perform evaluations with comparisons to solutions with the minimization on the makespan, i.e., the maximum completion time, under the given energy constraint. Simple extensions are made for the minimization of the weighted average flow time, but the optimality is not guaranteed.

The rest of this paper is organized as follows: Section 2 defines the problems. Section 3 presents some preliminary results and a motivational example. Section 4 considers processors with a continuous spectrum of the available speeds. The minimization of the average flow time for processors with a finite number of available speeds is then presented in Section 5. The extensions for the minimization of the average weighted flow time is shown in Section 6. Section 7 presents the performance evaluation of the proposed algorithms. Section 8 is the conclusion.

2. PROBLEM DEFINITIONS

We explore the scheduling of jobs on a dynamic voltage scaling (DVS) processor. The power consumption function $P()$ is defined as a function of the adopted processor speed s. The dynamic power consumption $P_\delta()$ resulting from the charging and discharging of gates on a DVS processor could be modeled as a function of the adopted processor speed s as follows [19]: $P_\delta(s) = C_{ef} V_{dd}^2 s$, where $s \propto \frac{(V_{dd}-V_t)^2}{V_{dd}}$, and C_{ef}, V_t, and V_{dd} denote the effective switch capacitance, the threshold voltage, and the supply voltage, respectively ($V_{dd} \geq V_t \geq 0$, and $C_{ef} > 0$). The dynamic power consumption function is a convex and increasing function of processor speeds. When $V_t = 0$, the dynamic power consumption function can be rephrased as a cubic function of the processor speed s. As reported in the literature [5, 11, 26], the dynamic power consumption function can be phrased as s^α, where α is a hardware-dependent factor. Leakage power consumption of the processor is assumed to be a non-negative constant. The power consumption function is the summation of the dynamic power consumption and the leakage power. For example, as shown in [7], the power consumption function could be $P(s) = s^3 + \beta$.

In this study, we consider two types of processors: (1) processors with a continuous spectrum of the available speeds between the upper-bounded speed $s_{\max}$ and the lower-bounded speed $s_{\min}$, and (2) processors with M distinct available speeds, say $s_1, s_2, \ldots, s_M$ indexed in an increasing order of the speeds. For brevity, we also denote s_1 by $s_{\min}$ and s_M by $s_{\max}$. Moreover, for the rest of this paper, the former type of processors is denoted by *ideal* processors while the latter type is denoted by *non-ideal* processors. Let $P(s)$ be the power consumption function of the processor under considerations at speed s. Note that the power consumption function of processor speed discussed in this paper can be *any function* that is strictly convex and increasing [16, 26], e.g., $P(s) \propto s^3$, where the convexity is defined in the field of non-negative real numbers. The number of CPU cycles executed in a time interval is linear in the processor speed. That is, the number of CPU cycles completed in time interval $(t_1, t_2]$ is $\int_{t_1}^{t_2} s(t)\mathrm{d}t$, where $s(t)$ is the processor speed at time t. The energy consumed in $(t_1, t_2]$ is $\int_{t_1}^{t_2} P(s(t))\mathrm{d}t$. Moreover, the time and energy overheads on speed (voltage) switching are assumed to be negligible. This is a common assumption in the literature [3, 5, 16, 18, 26]. Without loss of generality, the rest of the paper makes the physically reasonable assumption that the function $P(s)/s$ is strictly convex and monotonically increasing in $[s_{\min}, s_{\max}]$. For example, when $P(s) = s^3 + \beta$, $s_{\min}$ must be at least $\sqrt[3]{\frac{\beta}{2}}$ since $P(s)/s$ is minimized when $s = \sqrt[3]{\frac{\beta}{2}}$.

We are given a set $\mathbf{J} = \{J_1, J_2, \ldots, J_N\}$ of N jobs with the same arrival/ready time. Each given job J_i in $\mathbf{J}$ is specified by its computation requirement on the CPU as execution cycles c_i. A *schedule* of $\mathbf{J}$ is an assignment of processor speeds for the corresponding time intervals of the jobs in $\mathbf{J}$. The *total flow time* of a schedule is defined as the sum of the flow time of all the jobs. This paper considers the performance metric on the minimization of the *average flow time* of $\mathbf{J}$ under a given energy constraint E_b on a specified DVS-capable processor, where the average flow time of a schedule is the total flow time of $\mathbf{J}$ divided by N. For brevity, the arrival time of the these N jobs is 0, and hence, the average flow time for a schedule is the sum of the completion time of all the jobs in $\mathbf{J}$ divided by N. The problem is called the *energy-constrained average flow time minimization* problem.

A schedule is *feasible* for the energy-constrained average flow time minimization problem if the execution intervals of jobs are not overlapped, the energy constraint is satisfied, and the execution speeds are valid. An optimal schedule for the energy-constrained average flow time minimization problem is a schedule with the minimum average flow time among all feasible schedules. For the rest of this paper, we focus on input instances with $\sum_{J_i \in \mathbf{J}} P(s_{\min}) \cdot \frac{c_i}{s_{\min}} \leq E_b$, since there does not exist any feasible solution for the other cases. Moreover, if $\sum_{J_i \in \mathbf{J}} P(s_{\max}) \cdot \frac{c_i}{s_{\max}} \leq E_b$, executing all the jobs at speed $s_{\max}$ does not violate the energy constraint, and executing all the jobs at speed $s_{\max}$ consecutively in a non-decreasing order of the execution cycles of jobs is an optimal schedule. We consider the other cases for the rest of the paper.

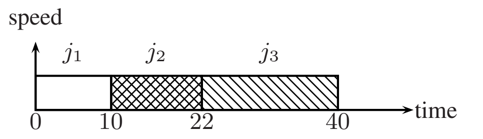
3. PRELIMINARY RESULTS AND A MOTIVATIONAL EXAMPLE

Because of the convexity of the power consumption function $P()$, in order to minimize the energy consumption of a job, the job must be executed at either one speed for ideal processors [26] or at most two consecutive speeds for non-ideal processors [12] for a fixed duration of executions. In other words, we only have to consider schedules that execute a job at one speed for ideal processors or at one or two consecutive speeds for non-ideal processors, since other types of schedules can be transformed into such a kind of schedule with the same average flow time and with less energy consumption.

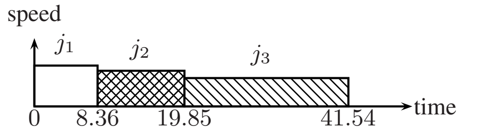
If the duration of executions for each job in $\mathbf{J}$ is determined, the minimization of the average flow time could be achieved by adopting the well-known shortest-job-first strategy [17, §3] by executing jobs in the non-decreasing order of their lengths of durations of executions. The following lemma states the optimality on the execution order of the jobs on their CPU execution cycles.

LEMMA 1. *There exists an optimal schedule which executes jobs in $\mathbf{J}$ in a non-decreasing order of their CPU execution cycles for both ideal and non-ideal processors.*

PROOF. For any optimal schedule, we know that (1) the processor executes some job at any moment from time 0 to the completion time of the job completed last, and (2) each job is executed one by one. Assume for contradiction that J_i is executed right after J_j



(a) Executing all the jobs in **J** at a common speed



(b) Executing j_1, j_2, and j_3 at speeds $0.598, 0.522$, and 0.415, respectively

Figure 1: A motivational example with 3 jobs.

in which $c_j > c_i$ in an optimal schedule S^* of **J** with the above two properties. The time at which job J_j starts (completes, respectively) is $t_{j,1}$ ($t_{j,2}$, respectively). $t_{i,1}$ and $t_{i,2}$ are defined similarly. We know $t_{j,2} = t_{i,1}$. Let $\hat{S}$ be a schedule with the same speed assignment as S^* all the time. $\hat{S}$ swaps the execution order of J_i and J_j so that J_i is executed right before J_j. Since $c_i < c_j$, $\hat{S}$ completes the execution of job J_i before $t_{j,2}$ and that of job J_j at speed $t_{i,2}$. As a result, the average flow time of $\hat{S}$ is less than that of S^*, since the flow time of all the jobs in $\mathbf{J} \setminus \{J_i, J_j\}$ remains. This contradicts the optimality of S^*. $\square$

By Lemma 1, for brevity, we index jobs in **J** so that $c_1 \leq c_2 \leq \cdots \leq c_N$. Based on Lemma 1, an intuitive solution is to minimize the maximum completion time of **J** under the given energy constraint with an execution order from J_1 to J_N one by one. For the scheduling on an ideal processor, such a solution executes all the jobs at a common speed, while two consecutive speeds are used for non-ideal processors in such a solution. Suppose that **J** consists of 3 jobs with $c_1 = 5, c_2 = 6$, and $c_3 = 9$. Consider the scheduling of **J** on an ideal processor. The ideal processor under considerations in this example is with $s_{\min} = 0.15$ and $s_{\max} = 1$. The power consumption function is $P(s) = s^3$ and the energy constraint is 5. Hence, the strategy to minimize the maximum completion time of **J** executes all the three jobs in **J** at speed 0.5 on the ideal processor. Executing these jobs in a shortest-job-first order at speed 0.5 leads to a solution with 24 time units on the average flow time with 5 unit of energy consumption, as shown in Figure 1(a). However, although executing all the jobs at a common speed minimizes the energy consumption when all the jobs have the same arrival time and share a common deadline [4], such an execution might not lead to solutions with minimum average flow time. Another speed setting by executing jobs (J_1, J_2, J_3) at speeds $(0.598, 0.522, 0.415)$ results in a solution with about 23.25 time units on average flow time and 4.973 unit of energy consumption, as shown in Figure 1(b).

4. IDEAL PROCESSORS

This section considers the *energy-constrained average flow time minimization* problem on an ideal processor. Suppose that the execution speed of job J_j is r_j. Executing jobs from J_1 to J_N consecutively leads to a solution, in which the flow time of job J_j is $\sum_{i=1}^{j} \frac{c_i}{r_i}$. Hence, the total flow time of such a solution is

$$\sum_{j=1}^{N}\sum_{i=1}^{j}\frac{c_i}{r_i} = \sum_{j=1}^{N}(N-j+1)\frac{c_j}{r_j},$$

while the energy consumption is $\sum_{j=1}^{N} P(r_j)\frac{c_j}{r_j}$.

Algorithm 1 : LM

```
Input: {J, E_b}
1: derive an optimal solution $(r_1^\dagger, r_2^\dagger, \ldots, r_{|\mathbf{J}|}^\dagger)$ for Equation (2);
2: if $r_1^\dagger > s_{\max}$ then
3:     return $(r_1^\dagger, \mathrm{LM}(\mathbf{J} \setminus \{J_1\}, E_b - P(s_{\max})\frac{c_1}{s_{\max}}))$;
4: else if $r_{|\mathbf{J}|}^\dagger < s_{\min}$ then
5:     return $(\mathrm{LM}(\mathbf{J} \setminus \{J_{|\mathbf{J}|}\}, E_b - P(s_{\min})\frac{c_{|\mathbf{J}|}}{s_{\min}}), r_{|\mathbf{J}|}^\dagger)$;
6: else
7:     return $(r_1^\dagger, r_2^\dagger, \ldots, r_{|\mathbf{J}|}^\dagger)$;
```

The following lemma shows that an optimal schedule will execute jobs at a non-increasing order of the processor speeds on an ideal processor.

LEMMA 2. *There exists an optimal schedule executing jobs (i) in a non-decreasing order of their CPU execution cycles and (ii) in a non-increasing order of the processor speeds on an ideal processor.*

PROOF. Assume for contradiction that S^* is an optimal schedule which executes job J_j at a lower speed than the speed of job J_{j+1}. Let r_j^* (r_{j+1}^*, respectively) be the speed to execute job J_j (J_{j+1}, respectively) in S^*. By definition, $s_{\min} \leq r_j^* < r_{j+1}^* \leq s_{\max}$. By executing job J_j at speed r_{j+1}^* and job J_{j+1} at speed $\frac{c_{j+1}}{\frac{c_j}{r_j^*} + \frac{c_{j+1}}{r_{j+1}^*} - \frac{c_j}{r_{j+1}^*}}$, both the energy consumption and the average flow time are reduced due to the convexity of the power consumption function, a contradiction. $\square$

Since $\sum_{j=1}^{N} P(s_{\min}) \cdot \frac{c_j}{s_{\min}} \leq E_b \leq \sum_{j=1}^{N} P(s_{\max}) \cdot \frac{c_j}{s_{\max}}$, the energy consumption of the optimal solution is E_b. Hence, the optimization problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{j=1}^{N}(N-j+1)\frac{c_j}{r_j} \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{N} P(r_j)\frac{c_j}{r_j} = E_b, \text{ and} \\ & s_{\min} \leq r_j \leq s_{\max}, \text{ for } j = 1, 2, \ldots, N. \end{aligned} \qquad (1)$$

Note that we index jobs in **J** so that $c_1 \leq c_2 \leq \cdots \leq c_N$.

By relaxing the last inequalities in Equation (1), we could have the following programming:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{j=1}^{N}(N-j+1)\frac{c_j}{r_j} \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{N} P(r_j)\frac{c_j}{r_j} = E_b, \end{aligned} \qquad (2)$$

which could be solved by applying the Lagrange Multiplier Method [20]. The optimal solution of Equation (2) must satisfy

$$\frac{\mathrm{d}((N-j+1)\frac{c_j}{r_j} - \lambda P(r_j)\frac{c_j}{r_j})}{\mathrm{d}r_j} = 0,$$

for all $j = 1, 2, \ldots, N$ for some constant λ. As a result, λ is $\frac{N-j+1}{-r_j(P'(r_j) - P(r_j)/r_j)}$, where $P'(s)$ is the first derivative of $P(s)$. For example, $\frac{r_i}{r_j} = \sqrt[\alpha]{\frac{N-i+1}{N-j+1}}$ and r_1 is $\sqrt[\alpha-1]{\frac{E_b}{\sum_{j=1}^{N} c_j (\frac{N-j+1}{N})^{\frac{\alpha-1}{\alpha}}}}$, when $P(s) = s^\alpha$. Hence, the optimal solution of Equation (2) could be derived efficiently.

If $s_{\min} = 0$ and $s_{\max} = \infty$, the above procedure derives an optimal solution. For general cases, by applying Lemma 2, we could find an index $j^* \geq 1$ by assigning the execution speed of $J_1, J_2, \ldots, J_{j^*-1}$ as $s_{\max}$ and an index $i^* \leq N$ by assigning the execution speed of $J_{i^*+1}, J_{i^*+2}, \ldots, J_N$ as $s_{\min}$ with $E_b^\dagger = E_b - \sum_{j=1}^{j^*-1} P(s_{\max})\frac{c_j}{s_{\max}} - \sum_{j=i^*+1}^{N} P(s_{\min})\frac{c_j}{s_{\min}} \geq 0$. The speeds of the other jobs $J_{j^*}, \ldots, J_{i^*}$ are obtained by solving the following programming:

$$\begin{array}{ll} \text{minimize} & \sum_{j=j^*}^{i^*}(N-j+1)\frac{c_j}{r_j} \\ \text{subject to} & \sum_{j=j^*}^{i^*} P(r_j)\frac{c_j}{r_j} = E_b^\dagger, \text{ and} \\ & r_j \geq 0, \text{ for } j = j^*, \ldots, i^*. \end{array} \tag{3}$$

Let $r_j^\dagger$ be the resulting speed assignment of job J_j in the set $\{J_{j^*}, \ldots, J_{i^*}\}$ of jobs by solving Equation (3). Our proposed algorithm, denoted as Algorithm LM and shown in Algorithm 1, finds the indices i^* and j^* so that

$$\frac{(N-i^*+1)}{(s_{\min})^2(\frac{P(s_{\min})}{s_{\min}})'} < \frac{(N-j+1)}{(r_j^\dagger)^2(\frac{P(r_j^\dagger)}{r_j^\dagger})'} < \frac{(N-j^*+1)}{(s_{\max})^2(\frac{P(s_{\max})}{s_{\max}})'}, \tag{4}$$

and $s_{\max} \geq r_j^\dagger \geq s_{\min}$ for any $j^* \leq j \leq i^*$. Then, $J_1, \ldots, J_{j^*-1}$ are executed at speed $s_{\max}$, $J_{i^*+1}, \ldots J_N$ are at speed $s_{\min}$, and J_j is executed at speed $r_j^\dagger$ for $j^* \leq j \leq i^*$.

Because $P(s)/s$ is a convex function, there must exist an index pair (j^*, i^*) satisfying Equation (4) with $s_{\max} \geq r_j^\dagger \geq s_{\min}$ for any $j^* \leq j \leq i^*$. The time complexity is $O(N^3)$, provided that the optimal solution of Equation (3), which is an extension of Equation (2), could be determined in $O(N)$. The time complexity could be reduced to $O(N^2 \log N)$ by performing a binary research. We omit the detail due to the space limitation. The optimality is shown as follows:

THEOREM 1. *Algorithm* LM *derives an optimal schedule of the energy-constrained average flow time minimization problem on any ideal processor.*

PROOF. We prove the lemma by applying the Karush-Kuhn-Tucker optimality condition for Equation (1) by finding a positive constant λ and three vectors $(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathbf{J}|})$, $(\hat\lambda_1, \hat\lambda_2, \ldots, \hat\lambda_{|\mathbf{J}|})$, and $(r_1^*, r_2^*, \ldots, r_{|\mathbf{J}|}^*)$ with

$$\begin{array}{ll} \frac{N-j+1}{(r_j^*)^2} - \lambda(\frac{P(r_j^*)}{r_j^*})' - \lambda_j + \hat\lambda_j = 0 & \\ s_{\min} \leq r_j^* \leq s_{\max}, \lambda_j \geq 0, \hat\lambda_j \geq 0 & \forall J_j \in \mathbf{J}, \text{ and} \\ (r_j^* - s_{\max})\lambda_j = 0, (s_{\min} - r_j^*)\hat\lambda_j = 0, & \\ \qquad \sum_{j=1}^N P(r_j^*)\frac{c_j}{r_j^*} = E_b. & \end{array} \tag{5}$$

It is clear that the resulting speed assignment $(r_1^\dagger, r_2^\dagger, \ldots, r_N^\dagger)$ of Algorithm LM satisfies $\sum_{j=1}^N P(r_j^\dagger)\frac{c_j}{r_j^\dagger} = E_b$ and $s_{\min} \leq r_j^\dagger \leq s_{\max}$.

By Algorithm LM, jobs $J_1, J_2, \ldots, J_{j^*-1}$ are executed at speed $s_{\max}$, where jobs $J_{i^*+1}, J_{i^*}, \ldots J_N$ are executed at speed $s_{\min}$. Let λ be $(N-j+1)/((r_j^\dagger)^2(\frac{P(r_j^\dagger)}{r_j^\dagger})')$ for some $j^* \leq j \leq i^*$. For any $j^* \leq j \leq i^*$, λ_j and $\hat\lambda_j$ are set as 0. Hence, the conditions in Equation (5) hold for $j^* \leq j \leq i^*$. For any $j < j^*$, let $\hat\lambda_j = 0$ and $\lambda_j = (N-j+1) - \lambda s_{\max}^2(\frac{P(s_{\max})}{s_{\max}})'$. By Equation (4), we know that $\lambda_j > 0$ for any $j < j^*$. Similarly, let $\lambda_j = 0$ and $\hat\lambda_j = \lambda s_{\min}^2(\frac{P(s_{\min})}{s_{\min}})' - (N-j+1) > 0$ for any $j > i^*$. The theorem is proved. □

5. NON-IDEAL PROCESSORS

This section copes with the *energy-constrained average flow time minimization* problem on a non-ideal processor. The proposed algorithm is a greedy algorithm based on a cost-benefit policy. First, we index jobs in $\mathbf{J}$ so that $c_1 \leq c_2 \leq \cdots \leq c_N$. Then, each job J_j is split into M *execution pieces*. The *additional energy consumption* $e_{j,m}$ of the m-th execution piece of job J_j is the difference of the energy consumption of J_j at speed s_m to that at speed s_{m-1}, i.e., $e_{j,m} = \frac{P(s_m)c_j}{s_m} - \frac{P(s_{m-1})c_j}{s_{m-1}}$, for $m = 2, 3, \ldots, M$. For brevity, $e_{j,1}$ is $P(s_1)c_j/s_1$. The *reduced total flow time* $\rho_{j,m}$ of the m-th execution piece of job J_j is the difference of the total flow time by executing job J_j at speed s_m instead of that at speed s_{m-1}, i.e., $\rho_{j,m} = (N-j+1)c_j(\frac{1}{s_{m-1}} - \frac{1}{s_m})$, for $m = 2, 3, \ldots, M$. For brevity, let $\rho_{j,1} = (N-j+1)\frac{c_j}{s_1}$. Since $P(s)/s$ is strictly convex and monotonically increasing in $[s_{\min}, s_{\max}]$, we know that

$$\frac{e_{j,m}}{\rho_{j,m}} < \frac{e_{j,m+1}}{\rho_{j,m+1}}, \tag{6}$$

for $m = 2, 3, \ldots, M-1$.

Initially, Algorithm GREEDY tries to execute all the jobs in $\mathbf{J}$ at speed s_1. That is, r_j is set as s_1 initially for all jobs J_j in $\mathbf{J}$. By $\sum_{J_i \in \mathbf{J}} P(s_{\min}) \cdot \frac{c_i}{s_{\min}} \leq E_b$, the initial speed assignment would result in a feasible schedule. Let Π be the set of the $N(M-1)$ execution pieces of the m-th execution pieces of job J_j for $m = 2, 3, \ldots, M$ and $j = 1, 2, \ldots, N$. Suppose that the additional energy consumption (the reduced total flow time, respectively) of execution piece π_i is e_i (ρ_i, respectively). We index the execution pieces in Π in a non-increasing order of the ratio of the reduced total flow time to the additional energy consumption of an execution piece. That is, $\frac{\rho_i}{e_i} \geq \frac{\rho_j}{e_j}$ for any $i < j$. For brevity, let $\psi(\pi_i)$ return job J_j and $\phi(\pi_i)$ return index m, where π_i is the m-th execution piece of job J_j. Algorithm GREEDY then greedily finds the minimum integer q such that

$$\sum_{j=1}^N \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q-1} e_i \leq E_b < \sum_{j=1}^N \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q} e_i.$$

For job J_j with $\psi(\pi_i) = J_j$ for some $1 \leq i < q$, r_j is updated to s_m, where m is $\max_{i=1,2,\ldots,q-1}\{\phi(\pi_i)|\psi(\pi_i) = J_j\}$. Suppose that $\psi(\pi_q)$ is J_{j^*} and $\phi(\pi_q)$ is m^*. Every job J_j in $\mathbf{J} \setminus \{J_{j^*}\}$ is executed at speed r_j, while $\frac{E_b - \left((\sum_{i=1}^{q-1} e_i) + \sum_{j=1}^N \frac{P(s_1)c_j}{s_1}\right)}{e_q}$ portion of job J_{j^*} is executed at speed $s_{\phi(\pi_q)}$, and the other portion of J_{j^*} is executed at speed r_{j^*}. The derived schedule then executes all the jobs in $\mathbf{J}$ from J_1 to J_N consecutively. Algorithm GREEDY is illustrated in Algorithm 2.

The time complexity of Algorithm GREEDY is shown as follows: The sorting and derivation of execution pieces could be done in $O(NM \log(NM))$ time. The time for the remaining part is $O(NM)$. The overall time is $O(NM \log(NM))$.

Algorithm 2 : GREEDY

```
Input: ${\mathbf{J}, E_b}$
1: if $\sum_{j=1}^N \frac{P(s_1)c_j}{s_1} > E_b$ then
2:     return "no feasible schedule exists";
3: else if $\sum_{j=1}^N \frac{P(s_M)c_j}{s_M} \leq E_b$ then
4:     execute jobs from $J_1$ to $J_N$ at speed $s_M$;
5: else
6:     $r_j = s_1$, for $j = 1, 2, \ldots, N$;
7:     let $\Pi$ be the set of the $N(M-1)$ execution pieces of the $m$-th execution pieces of job $J_j$ for $m = 2, 3, \ldots, M$ and $j = 1, 2, \ldots, N$;
8:     sort execution pieces in $\Pi$ in a non-increasing order of $\frac{\rho_i}{e_i}$ of execution piece $\pi_i$;
9:     find the minimum $q$ such that $\sum_{j=1}^N \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q-1} e_i \leq E_b < \sum_{j=1}^N \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q} e_i$;
10:    for each job $J_j$ with $\psi(\pi_i) = J_j$ for some $1 \leq i < q$, let $r_j$ be $s_m$, where $m$ is $\max_{i=1,2,\ldots,q-1}\{\phi(\pi_i)|\psi(\pi_i) = J_j\}$;
11:    $J_{j^*} \leftarrow \psi(\pi_q)$;
12:    execute jobs from $J_1$ to $J_N$, where every job $J_j \in \mathbf{J} \setminus \{J_{j^*}\}$ is executed at speed $r_j$, $\frac{E_b - \left((\sum_{i=1}^{q-1} e_i) + \sum_{j=1}^N \frac{P(s_1)c_j}{s_1}\right)}{e_q}$ portion of job $J_{j^*}$ is executed at $s_{\phi(\pi_q)}$, and the other portion of $J_{j^*}$ is executed at speed $r_{j^*}$;
```

The feasibility of the derived schedule is shown as follows: Because of the inequality in Equation (6), when the m-th execution piece of job J_j is one of the first q execution pieces in Π, the k-th execution pieces of job J_j is also one of the first q execution pieces in Π for any $1 < k < m$. The energy consumption to execute job J_j at the updated r_j is $\sum_{j=1}^{N} \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q-1} e_i$, which is no more than E_b by the definition of q. The execution of job J_{j^*} at speed $s_{\phi(\pi_q)}$ consumes $E_b - (\sum_{j=1}^{N} \frac{P(s_1)c_j}{s_1} + \sum_{i=1}^{q-1} e_i)$ additional energy. As a result, the derived schedule is feasible.

The optimality of Algorithm GREEDY is as follows:

THEOREM 2. *Algorithm* GREEDY *derives an optimal schedule of the energy-constrained average flow time minimization problem on any non-ideal processor.*

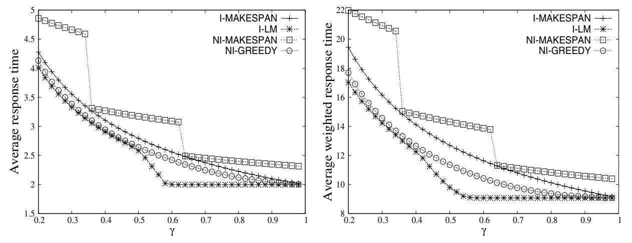
PROOF. From the convexity of the power consumption, a job must be executed at one or two consecutive speeds [12] to minimize the energy consumption. Based on Lemma 1, we only have to consider schedules that execute jobs from J_1 to J_N. For any schedule that does not satisfy the above two properties, the schedule can be transformed into another one with the two properties.

For any feasible schedule S with the above properties, we show that the total flow time is no less than the total flow time of the derived schedule S_g of Algorithm GREEDY. We divide jobs in $\mathbf{J}$ into two job sets $\mathbf{J}^1$ and $\mathbf{J}^2$, in which each of the jobs in $\mathbf{J}^1$ is executed at one speed and each of the jobs in $\mathbf{J}^2$ is executed at two speeds in S. For job J_j in $\mathbf{J}^1$, $r_j^\dagger$ is the speed that J_j is executed at in S. For job J_j in $\mathbf{J}^2$, $r_j^\dagger$ is the lower speed that J_j is executed at in S, and $x_j^\dagger$ is the portion that J_j is executed at speed $r_j^\dagger$. By the definition of S, we know that the remaining $1 - x_j^\dagger$ portion of job J_j is executed at the higher consecutive speed of $r_j^\dagger$. For brevity, let $x_j^\dagger = 1$ for any job J_j in $\mathbf{J}^1$. Let Z^S be a vector of $z_{j,m}^S$ for $j = 1, 2, \ldots, N$ and $m = 1, 2, \ldots, M$. If r_j is s_m, $z_{j,1}^S, z_{j,2}^S, \ldots, z_{j,m}^S$ are set as 1, $z_{j,m+1}^S$ is set as $1 - x_j^\dagger$, and $z_{j,m+2}^S, z_{j,m+3}^S, \ldots, z_{j,M}^S$ are set as 0. By the definition of execution pieces, we know that the energy consumption of S is equal to $\sum_{j=1}^{N}\sum_{m=1}^{M} e_{j,m} z_{j,m}^S$, and the total flow time of S is $(\sum_{j=1}^{N} \rho_{j,1}) - (\sum_{j=1}^{N}\sum_{m=1}^{M} \rho_{j,m} z_{j,m}^S)$. The vector Z^{S_g} is defined similarly for schedule S_g. Since execution pieces are sorted in a non-increasing order of the ratio of the reduced total flow time to the additional energy consumption of an execution piece, we know that $\sum_{j=1}^{N}\sum_{m=1}^{M} \rho_{j,m} z_{j,m}^S \leq \sum_{j=1}^{N}\sum_{m=1}^{M} \rho_{j,m} z_{j,m}^{S_g}$. As a result, the average flow time of S is no less than that of S_g. □

6. ENERGY-CONSTRAINED AVERAGE WEIGHTED FLOW TIME

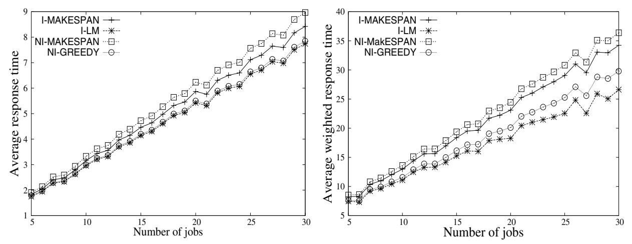
Simple extensions of our proposed algorithms could be made for pursuing the minimization of the average weighted flow time, in which the weighted flow time of a job is defined as the flow time times the given weight of the job. If the duration of executions for each job is determined, the minimization of the average weighted flow time could be achieved by applying the well-known weighted-shortest-job-first strategy [17, §3], which executes jobs in the non-decreasing order of their lengths of durations of executions divided by their weights. By executing jobs in a non-decreasing order of their execution cycles divided by their weights, we could revise Algorithm LM and Algorithm GREEDY to derive a solution for the minimization of the average weighted flow time. Let w_j be the weight of job J_j. With the above order, we know $\frac{c_i}{w_i} \leq \frac{c_j}{w_j}$ when $i < j$.

Algorithm LM is modified by taking the objective function of Equation (1) as $\sum_{j=1}^{N}(\sum_{i=j}^{N} w_i)\frac{c_j}{r_j}$. The value $N - j + 1$ in Section 4 is replaced by $\sum_{i=j}^{N} w_i$, and the algorithm still works. Algorithm GREEDY is modified by taking $\rho_{j,m}$ as $(\sum_{i=j}^{N} w_i)c_j(\frac{1}{s_{m-1}} - \frac{1}{s_m})$.



(a) unweighted cases (b) weighted cases

Figure 2: Experimental results for fixed job sets when the number of jobs is 10



(a) unweighted cases (b) weighted cases

Figure 3: Experimental results when γ **is** 0.4

The above two revised algorithms can be proved to be optimal under a specified execution order. However, the weighted-shortest-job-first strategy does not result in optimal solutions neither for ideal nor for non-ideal processors. Consider two jobs J_1 and J_2 with $c_1 = 1$, $c_2 = 10$, $w_1 = 1 + \epsilon$, and $w_2 = 10$ on a non-ideal processor with $s_1 = 0.5, s_2 = 1$, and $P(s) = s^3$, where E_b is 10.25 and $\epsilon < 0.5$. Executing the two jobs in a non-decreasing order of the ratios of their execution cycles divided by their weights, i.e., J_1 before J_2, results in a solution with $(121 + \epsilon)/2$ average weighted flow time, and vice versa with $(112 + 12\epsilon)/2$ average weighted flow time. For ideal processors with two jobs, in which $w_1 = 0.51$, $w_2 = 1$, $c_1 = 0.5$, $c_2 = 1$, $s_{\max} = 1$, $s_{\min} = 0.9$, and $E_b = 1.405$, executing J_2 before J_1 results in a solution with $0.89\bar{6}$ average weighted flow time by executing J_2 at speed 1 and J_1 at speed 0.9, and vice versa with about 0.9031 average weighted flow time by executing J_1 at speed 1 and J_2 at speed 0.951315.

7. PERFORMANCE EVALUATION

We compare the performance of Algorithm LM and Algorithm GREEDY with solutions derived from Algorithm MAKESPAN, in which Algorithm MAKESPAN decides the effective speed for $\mathbf{J}$ to minimize the maximum completion time under the given energy constraint and executes jobs in the shortest-job-first order. For non-ideal processors Algorithm MAKESPAN executes jobs at two consecutive speeds from the higher one to the lower one. Algorithm MAKESPAN is optimal when the objective is on the minimization of the (maximum) job completion time. We denote such an algorithm as Algorithm I-MAKESPAN (NI-MAKESPAN, respectively) when ideal (non-ideal, respectively) processors are considered. Since Algorithm GREEDY is designed for non-ideal processors, we also denote it as Algorithm NI-GREEDY, while Algorithm LM is denoted as I-LM.

We perform evaluations for non-ideal processors by taking the Intel XScale processor [24] as an example, in which there are five processor speeds: 150, 400, 600, 800, and 1000 MHz with power consumption 80, 170, 400, 900, and 1600 mWatt. We normalize the processor speed so that the maximum speed is 1 and the minimum speed is 0.15. For ideal processors, the processor speed is continuous in the speed range of the Intel XScale, i.e., the speed is in $[0.15, 1]$. The power consumption function of Intel XScale for ideal processors is approximated as $P(s) = 0.08 + 1.52s^3$ Watt. Other settings on the power consumption function could have very similar results. For each job J_j, c_j is normalized as a random variable in $(0, 1]$. For the evaluations of the minimization of the average weighted flow time, the weight of a job is a random variable in $(0.1, 10.1]$.

For a given job set $\mathbf{J}$ with N jobs, let $E_{\max}$ ($E_{\min}$, respectively) be the energy consumption by executing all of the jobs at speed $s_{\max}$ ($s_{\min}$, respectively). The energy consumption constraint E_b for a job set depends on the value $0 \leq \gamma \leq 1$. For a specified γ, E_b is set as $E_{\min} + \gamma(E_{\max} - E_{\min})$. We simulate job sets with 10 jobs by varying γ from 0.1 to 0.9. Another evaluation is done for job sets with 5 to 30 jobs by taking γ as 0.4. For each configuration, we perform evaluations with 99% confidence interval. The *average (weighted) flow time* is taken as the performance metric in the experiments.

Figure 2 shows the average flow time and weighted average flow time for the evaluated algorithms when there are 10 jobs by varying γ from 0.2 to 0.98, stepped by 0.02. For a fixed job set, we vary the energy constraint. As a result, the greater the value of energy constraint, the less the average (weighted) flow time for any of the evaluated algorithms. Algorithm I-LM outperforms Algorithm I-MAKESPAN, and Algorithm NI-GREEDY outperforms Algorithm NI-MAKESPAN. The significant performance improvement from $\gamma = 0.34$ to $\gamma = 0.36$ and from $\gamma = 0.62$ to $\gamma = 0.64$ is because the resulting schedule from Algorithm NI-MAKESPAN will execute most jobs at one available speed which is close to the resulting schedule from Algorithm I-MAKESPAN. When the energy constraint is great enough, i.e., $\gamma \geq 0.6$, the performance of Algorithm I-LM becomes steady. This comes from Lemma 2 since the resulting schedule will execute most jobs at speed $s_{\max}$. The more amount of energy for executions only has little improvement for the average (weighted) flow time.

Figure 3 shows the average flow time and weighted average flow time for the evaluated algorithms when γ is 0.4 by varying the number of jobs from 5 to 30. Similarly, Algorithm I-LM outperforms Algorithm I-MAKESPAN, and Algorithm NI-GREEDY outperforms Algorithm NI-MAKESPAN.

8. CONCLUSION

This paper explores the minimization of the average flow time under a given energy constraint on a DVS processor. Two algorithms are proposed to derive optimal solutions for processors with a continuous spectrum of the available speeds or with a finite number of discrete speeds. Jobs are executed in a non-decreasing order of their execution cycles. The proposed algorithms are also evaluated with comparisons to solutions which execute every job at a common (effective) speed. We also provide extensions for the minimization of the average weighted flow time. Simulation results show the effectiveness of the proposed algorithms. It is still open to minimize the average weighted flow time under a given energy constraint.

For future research, we will explore the minimization of the average flow time for jobs with different arrival times. It is also interesting to derive algorithms to minimize the average weighted flow time under a given energy constraint.

References

[1] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. In *STACS*, pages 621–633, 2006.

[2] T. A. Alenawy and H. Aydin. Energy-constrained scheduling for weakly-hard real-time systems. In *Proceedings of the 26th IEEE Real-time Systems Symposium (RTSS'05)*, pages 376–385, 2005.

[3] 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*, pages 225–232, 2001.

[4] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 95–105, 2001.

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

[6] J.-J. Chen and T.-W. Kuo. Voltage-scaling scheduling for periodic real-time tasks in reward maximization. In *the 26th IEEE Real-Time Systems Symposium (RTSS)*, pages 345–355, 2005.

[7] J.-J. Chen and T.-W. Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 153–162, 2006.

[8] J.-J. Chen, T.-W. Kuo, and H.-I. Lu. Power-saving scheduling for weakly dynamic voltage scaling devices. In *Workshop on Algorithms and Data Structures (WADS)*, pages 338–349, 2005.

[9] J.-J. Chen, T.-W. Kuo, and C.-S. Shih. 1+ϵ approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor. In *the 2nd ACM Conference on Embedded Software (EMSOFT)*, pages 247–250, 2005.

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

[11] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 37–46, 2003.

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

[13] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, pages 275–280, 2004.

[14] W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proceedings of the 40th Design Automation Conference*, pages 125–130, 2003.

[15] Y.-H. Lee, K. P. Reddy, and C. M. Krishna. Scheduling techniques for reducing leakage power in hard real-time systems. In *15th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 105–112, 2003.

[16] 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.

[17] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Precentice Hall, 2nd edition, 2002.

[18] K. Pruhs, P. Uthaisombut, and G. J. Woeginger. Getting the best response for your erg. In *9th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 14–25, 2004.

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

[20] R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

[21] C. Rusu, R. Melhem, and D. Mosse. Maximizing the system value while satisfying time and energy constraints. In *IEEE 23th Real-Time System Symposium*, pages 246–255, Dec. 2002.

[22] C. Rusu, R. Melhem, and D. Mossé. Multiversion scheduling in rechargeable energy-aware real-time systems. In *EuroMicro Conference on Real-Time Systems (ECRTS'03)*, pages 95–104, 2003.

[23] INTEL. Strong ARM SA-1100 Microprocessor Developer's Manual, 2003. INTEL.

[24] INTEL-XSCALE, 2003. http://developer.intel.com/design/xscale/.

[25] TRANSMETA, 2003.

[26] 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.