

Power-Saving Scheduling for Weakly Dynamic Voltage Scaling Devices*

Jian-Jia Chen¹, Tei-Wei Kuo², and Hsueh-I Lu²

¹ Department of Computer Science and Information Engineering
National Taiwan University, Taiwan, Republic of China.
Email:r90079@csie.ntu.edu.tw

² Department of Computer Science and Information Engineering
Graduate Institute of Networking and Multimedia
National Taiwan University, Taiwan, Republic of China.
Email:{ktw,hil}@csie.ntu.edu.tw

Abstract. We study the problem of non-preemptive scheduling to minimize energy consumption for devices that allow dynamic voltage scaling. Specifically, consider a device that can process jobs in a non-preemptive manner. The input consists of (i) the set R of available speeds of the device, (ii) a set J of jobs, and (iii) a precedence constraint Π among J . Each job j in J , defined by its arrival time a_j , deadline d_j , and amount of computation c_j , is supposed to be processed by the device at a speed in R . Under the assumption that a higher speed means higher energy consumption, the power-saving scheduling problem is to compute a feasible schedule with speed assignment for the jobs in J such that the required energy consumption is minimized.

This paper focuses on the setting of *weakly* dynamic voltage scaling, i.e., speed change is not allowed in the middle of processing a job. To demonstrate that this restriction on many portable power-aware devices introduces hardness to the power-saving scheduling problem, we prove that the problem is NP-hard even if $a_j = a_{j'}$ and $d_j = d_{j'}$ hold for all $j, j' \in J$ and $|R| = 2$. If $|R| < \infty$, we also give fully polynomial-time approximation schemes for two cases of the general NP-hard problem: (a) all jobs share a common arrival time, and (b) $\Pi = \emptyset$ and for any $j, j' \in J$, $a_j \leq a_{j'}$ implies $d_j \leq d_{j'}$. To the best of our knowledge, there is no previously known approximation algorithm for any special case of the NP-hard problem.

1 Introduction

With the increasing popularity of portable systems, energy efficiency has become a major design issue in hardware and software implementations [4, 9, 16, 21, 22, 27]. Power-aware resource management for portable devices is a critical design factor since most of them are driven by their own power sources

* Support in parts by research grants from ROC National Science Council NSC-93-2752-E-002-008-PAE.

(e.g., batteries). Energy-efficient electronic circuit designs, e.g., [1,34], were proposed in the past decade, and various vendors have provided processors, memory chips, storage devices, or even motherboards equipped with the voltage scaling technology. For example, the flash-memory chip designed by the Intel Corp. [12] supports several voltage levels for operations. Because of the characteristics of many similar storage devices, the supply voltage for an I/O operation remains unchanged for the entire duration of the operation. Non-preemptivity in operation scheduling is also the inherent nature of many I/O devices. Besides, for performance-sensitive devices, the balance between performance and energy consumption must be taken into considerations. We consider power savings on a device capable of supporting several levels of supply voltages with predictable execution times and energy consumption to process jobs.

Let X_R be a device that processes jobs one at a time in a non-preemptive manner, where R consists of the available speeds of X_R . As restricted for most portable devices, speed change is not allowed in the middle of processing a job, e.g., in the flash-memory chip [12]. The *energy-consumption rate* of X_R is a function ϕ over R such that the energy required for X_R to process a job at speed $r \in R$ for t time units is $\phi(r) \cdot t$. The rest of the paper makes the physically reasonable assumption that the function $\phi(r)/r$ is monotonically increasing, i.e., $r > r'$ implies $\frac{\phi(r)}{r} > \frac{\phi(r')}{r'}$.

For any set S , let $|S|$ denote the cardinality of S . All numbers throughout the paper are rational. Let J consist of jobs $1, 2, \dots, |J|$ to be processed by X_R . For each $j \in J$, let c_j be the time required for X_R to process job j at speed 1, let d_j be the deadline for completing job j , and let a_j be the arrival time for job j . If the precedence constraint Π is present on J , a schedule cannot execute job j' before j when j' is a successor to j . It is reasonable to assume that $d_j \leq d_{j'}$ if j' is a successor to j . For notational brevity, let $d_1 \leq d_2 \leq \dots \leq d_{|J|}$. If $d_j = d_{j'}$ and $a_j < a_{j'}$, then $j < j'$. Without loss of generality, we assume $\min_{j \in J} a_j = 0$. A schedule s for J is *feasible* if each job $j \in J$ is assigned a speed $s_j \in R$ and processed after a_j and before d_j without speed change, preemption, or violating the precedence constraints. We say that s is an *earliest-deadline-first schedule* for J (with respect to X_R) if each job $j \in J$ is processed by X_R before jobs $j+1, \dots, |J|$ as early as possible. It is well known (e.g., [8, §A5.1]) that if the jobs in J have arbitrary arrival times and deadlines, determining whether the jobs in J can be scheduled to meet all deadlines is strongly NP-complete even if $|R| = 1$. The time required for X_R to process job j at speed $r \in R$ is assumed being c_j/r . The *energy consumption* $\Phi(s)$ of a schedule s is $\sum_{j \in J} \phi(s_j) c_j / s_j$. The POWER-SAVING SCHEDULING problem is to find a feasible schedule s for J with minimum $\Phi(s)$. Clearly, if the energy-consumption rate of X_R is linear, which is highly unlikely in practice, then any schedule for J has the same energy consumption.

Regardless of the property of the energy consumption function, it is not hard to show that the POWER-SAVING SCHEDULING problem does not admit any polynomial-time approximation algorithm unless $P = NP$, when jobs have arbitrary arrival times and deadlines. Assuming that there exists a polynomial-time

approximation algorithm \mathcal{K} for the POWER-SAVING SCHEDULING problem, algorithm \mathcal{K} can be used to solve the following NP-complete 3-PARTITION problem [8] in polynomial time: Given a set A of $3M$ elements, a bound $B \in \mathbb{Z}^+$, and a size $w(a) \in \mathbb{Z}^+$ for each $a \in A$, where $B/4 < w(a) < B/2$ and $\sum_{a \in A} w(a) = MB$, the 3-PARTITION problem is to find a partition of A into M disjoint sets A_1, A_2, \dots, A_M such that $\sum_{a \in A_j} w(a) = B$ for $1 \leq j \leq M$. For each element $a \in A$, a unique job j is created by setting $a_j = 0$, $d_j = (M + 1)B - 1$, and $c_j = s_{|R|} \cdot w(a)$. Another job set J' is also constructed by creating $M - 1$ jobs, where $a_j = (j + 1)B - 1$, $d_j = (j + 1)B$, and $c_j = s_{|R|}$ for the j -th job in J' . It is clear that there exists a feasible schedule for the resulting job set above if and only if there exists a partition for the 3-PARTITION problem. Since \mathcal{K} is a polynomial-time approximation algorithm the POWER-SAVING SCHEDULING problem, \mathcal{K} can be applied to determine the 3-PARTITION problem in polynomial time by examining the feasibility of the derived schedule of \mathcal{K} . This contradicts the assumption that $P \neq NP$.

Our contribution We investigate the intractability of the problem. Moreover, we give a fully polynomial-time approximation scheme for two practically important cases: (a) all jobs share a common arrival time, and (b) $\Pi = \emptyset$ and for any $j, j' \in J$, $a_j \leq a_{j'}$ implies $d_j \leq d_{j'}$ ($a_j \leq a_{j'}$ if $j < j'$).

Let *basic case* stand for the situation that $|R| = 2$ and that all jobs in J have a common deadline d and a common arrival time 0. We show that the POWER-SAVING SCHEDULING problem is NP-complete even for the basic case. The hardness comes from the dis-allowance of speed change on devices with a finite set of speeds to select from. Moreover, we give a fully polynomial-time approximation scheme for the POWER-SAVING SCHEDULING problem when (a) all jobs share a common arrival time, and (b) $\Pi = \emptyset$ and for any $j, j' \in J$, $a_j \leq a_{j'}$ implies $d_j \leq d_{j'}$, based upon standard techniques of dynamic programming and rounding. Specifically, we first show that a special case of the problem can be solved in pseudopolynomial-time algorithm by a dynamic program. (We comment that the dynamic programs studied by Chen, Lu, and Tang [5] would have sufficed if their schedules were not allowed to violate deadlines.) To turn a pseudopolynomial-time algorithm into a fully polynomial-time approximation scheme via rounding, we need the following lemma to obtain a good estimate of the minimum energy consumption.

Lemma 1 (Yao, Demers, and Shenker [35]). *Let R be the set of non-negative rational numbers. Then, given a set of jobs with arbitrary arrival times and deadlines, a feasible schedule for J on X_R allowing preemption with minimum energy consumption can be solved in $O(|J| \log^2 |J|)$ time.*

Comment: the result stated in Lemma 1 requires that the energy consumption rate function ϕ is convex and increasing, which is implied by the global assumption of the present paper that $\phi(r)/r$ is monotonically increasing.

Related work In [14], a 3-approximation algorithm is proposed for off-line job scheduling, when processors have a special state *sleep*. If a processor has multi-

ple special states beside *running*, (e.g., *idle*, *sleep*, and *standby*), processors could consume much less energy while no job is processing. Although special states provide more flexibility for energy-aware job scheduling, overheads on energy consumption and time latencies for state transitions must be considered. The on-line competitive algorithm proposed by Yao, et al. [35] is extended to handle processors with a single *sleep* state in [14] or multiple special states in [13].

Ishihara and Yasuura [15] show that the minimum energy consumption problem can be formulated as an integer linear program, when $|R|$ is finite, speed change is allowed, and all jobs have the same arrival time and deadline. They show that there exists an optimal schedule with at most two processor speeds and at most one job is processed at two different processor speeds. However the proofs in [15] consider only a specific processor model which is proposed in [1, 34]. In [3], the results in [15] are extended for any convex function. Energy-efficient scheduling has been extensively studied in off-line, e.g., [10, 19, 26], or on-line, e.g., [17, 23–25, 28–31], fashions. Algorithms considering time-cost trade-offs on scheduling are proposed in [6, 7, 32, 33].

The results on minimization of energy consumption on processors could not be applied directly to I/O devices. Generally, processors can process jobs in a preemptive manner. In contrast, I/O devices might perform operations in a non-preemptive manner, and no speed change is allowed in the middle of processing an operation. Chang, Kuo, and Lo [2] propose an adjustment mechanism to dynamically adjust the supply voltage for a flash memory to reduce the energy consumption. The proposed heuristic algorithm is efficient but the optimality is not proved. Hong, Kirovski, Qu, Potkonjak, and Srivastava [10] propose a heuristic algorithm for minimization of the energy consumption on a non-preemptive device, where the supply voltages of the device are available between two given positive thresholds. Besides, Manzak and Chakrabarti [20] consider a minimum energy consumption problem on a system equipped with non-preemptive I/O devices capable of supporting multiple levels of voltages, where voltage switch introduces overheads in energy and time. On-line algorithms are proposed to minimize the energy consumption for executions of real-time tasks, provided that there is a given feasible schedule as an input. Although the derived schedules are feasible and power savings were achieved in the experimental results, the optimality is not shown. To the best of our knowledge, no previous scheduling algorithm to minimize the energy consumption for X_R is known with theoretical analysis of optimality for energy consumption or power savings.

For the remainder of the paper, define $R = \{r_1, r_2, \dots, r_{|R|}\}$ with $r_1 < r_2 < \dots < r_{|R|}$. The rest of the paper is organized as follows. Section 2 addresses the basic case. Section 3 shows our approximation scheme. Section 4 concludes this paper.

2 Basic case

In this section, we show that finding a schedule with minimum energy consumption is NP-complete for the basic case. We then present a fully polynomial-time approximation scheme for the basic case as a warm-up. For any subset I of J , let $s(I)$ be the schedule s for J with $s_i = r_1$ for each $i \in I$ and $s_i = r_2$ for each $i \in J - I$. Clearly, we have

$$\Phi(s(I)) = \sum_{i \in I} \phi(r_1)c_i/r_1 + \sum_{i \in J-I} \phi(r_2)c_i/r_2.$$

Theorem 1. *The POWER-SAVING SCHEDULING problem for X_R is NP-complete even for the basic case.*

Proof. It is clear that the POWER-SAVING SCHEDULING problem is in NP. It suffices to show the NP-hardness by a reduction from the following NP-complete SUBSET SUM problem [8]: Given a number w_j for each index $j \in J$ and another arbitrary number w , the problem is to determine whether there is a subset I of J with $\sum_{i \in I} w_i = w$.

An instance for the basic case of the POWER-SAVING SCHEDULING problem is constructed as follows: For each index $j \in J$, we create a job j with $c_j = r_1 r_2 w_j$. Let the common deadline d be $w(r_2 - r_1) + r_1 \sum_{j \in J} w_j$. Clearly, the set J of jobs is feasible. Also, for any subset I of J , we have

$$\Phi(s(I)) = r_2 \phi(r_1) \sum_{i \in I} w_i + r_1 \phi(r_2) \sum_{i \in J-I} w_i = (r_2 \phi(r_1) - r_1 \phi(r_2)) \sum_{i \in I} w_i + r_1 \phi(r_2) \sum_{j \in J} w_j. \quad (1)$$

We show that there is a set $I \subseteq J$ with $\sum_{i \in I} w_i = w$ if and only if the set J of jobs admits a feasible schedule s with $\Phi(s) = w(r_2 \phi(r_1) - r_1 \phi(r_2)) + r_1 \phi(r_2) \sum_{j \in J} w_j$. As for the if-part, let I consist of the jobs i with $s_i = r_1$, which implies $s(I) = s$. By Equation (1) and $\Phi(s) = w(r_2 \phi(r_1) - r_1 \phi(r_2)) + r_1 \phi(r_2) \sum_{j \in J} w_j$, we know $(r_2 \phi(r_1) - r_1 \phi(r_2)) \sum_{i \in I} w_i = w(r_2 \phi(r_1) - r_1 \phi(r_2))$. Since $r_2 \phi(r_1) < r_1 \phi(r_2)$, we have $\sum_{i \in I} w_i = w$. As for the only-if-part, let $s = s(I)$. One can easily see the feasibility of s by verifying that $\sum_{i \in I} w_i = w$ implies $\sum_{i \in I} c_i/r_1 + \sum_{i \in J-I} c_i/r_2 = r_2 \sum_{i \in I} w_i + r_1 \sum_{i \in J-I} w_i = d$. By $\sum_{i \in I} w_i = w$ and Equation (1), we have $\Phi(s) = w(r_2 \phi(r_1) - r_1 \phi(r_2)) + r_1 \phi(r_2) \sum_{j \in J} w_j$. \square

Given a number w_j for each index $j \in J$ and another arbitrary number w , the MAXIMUM SUBSET SUM problem is to find a subset I of J with $\sum_{i \in I} w_i \leq w$ such that $\sum_{i \in I} w_i$ is maximized. For the rest of the section, we show how to obtain a fully polynomial-time approximation scheme for the basic case of the POWER-SAVING SCHEDULING problem based upon the following lemma.

Lemma 2 (Ibarra and Kim [11]). *The MAXIMUM SUBSET SUM problem admits a fully polynomial-time $\frac{1}{(1-\delta)}$ -approximation algorithm $\text{SUBSET}(w_1, w_2, \dots, w_{|J|}, w, \delta)$ for any $0 < \delta < 1$.*

Given the algorithm shown in Algorithm 1, we have the following lemma.

Algorithm 1

Input: J, r_1, r_2, d, ϵ ;

Output: A feasible schedule s with almost minimum energy consumption;

1: let $c = r_1(r_2d - \sum_{j \in J} c_j)/(r_2 - r_1)$;

2: let I be the subset returned by $\text{SUBSET}(c_1, c_2, \dots, c_{|J|}, c, \delta)$, where $\delta = \frac{\epsilon r_2 \phi(r_1)}{r_1 \phi(r_2)}$;

3: output $s(I)$;

Lemma 3. *Algorithm 1 is a $(1 + \epsilon)$ -approximation for the basic case of the POWER-SAVING SCHEDULING problem for any $\epsilon > 0$.*

Proof. Let I^* be a subset of J with $\sum_{i \in I^*} c_i \leq c$ such that $\sum_{i \in I^*} c_i$ is maximized. By the choice of c , one can verify that both $s(I)$ and $s(I^*)$ are feasible schedules for J . Moreover, $s(I^*)$ is an optimal schedule. We show $\Phi(s(I)) \leq (1 + \epsilon)\Phi(s(I^*))$ as follows. By Lemma 2, we have $\sum_{i \in I} c_i \leq \sum_{i \in I^*} c_i \leq \sum_{i \in I} c_i/(1 - \delta)$. It follows that

$$\begin{aligned} \Phi(s(I)) - \Phi(s(I^*)) &= \phi(r_1) \left(\sum_{i \in I} c_i - \sum_{i \in I^*} c_i \right) / r_1 + \phi(r_2) \left(\sum_{i \in I^*} c_i - \sum_{i \in I} c_i \right) / r_2 \\ &\leq \phi(r_2) \left(\sum_{i \in I^*} c_i - \sum_{i \in I} c_i \right) / r_2 \leq \delta \sum_{i \in I^*} \phi(r_2) \cdot c_i / r_2 \\ &= \epsilon \sum_{i \in I^*} \phi(r_1) \cdot c_i / r_1 \leq \epsilon \cdot \Phi(s(I^*)). \end{aligned}$$

The lemma is proved. \square

3 Our approximation scheme

Recall that $R = \{r_1, r_2, \dots, r_{|R|}\}$, where $r_1 < r_2 < \dots < r_{|R|}$. Define

$$\gamma = \max_{2 \leq i \leq |R|} \frac{r_{i-1} \cdot \phi(r_i)}{r_i \cdot \phi(r_{i-1})}.$$

An execution sequence is said to be optimal if any feasible schedule can be translated into such a sequence without increasing the energy consumption. Let s^* be a feasible schedule s for the input job set J with minimum $\Phi(s)$. We propose our approximation scheme based on the following lemma which ignores the precedence constraint Π first.

Lemma 4. *Suppose that we are given a schedule \hat{s} satisfying $\Phi(s^*) \leq \Phi(\hat{s}) \leq \gamma \Phi(s^*)$, the earliest-deadline-first execution sequence is optimal, and $\Pi = \emptyset$, then it takes $O(|R||J|^2(\epsilon^{-1} + \log \gamma))$ time and $O(\epsilon^{-1}|R||J|^2)$ space to compute a $(1 + \epsilon)$ -optimal solution for the POWER-SAVING SCHEDULING problem for any parameter $0 < \epsilon \leq 1$.*

Proof. Our approximation scheme is based upon the standard rounding technique. For each $j \in J$ and each $r \in R$, let $\psi(j, r)$ denote the energy consumption $\phi(r)c_j/r$ required by X_R for processing job j at speed r . That is,

$\Phi(s) = \sum_{j \in J} \psi(j, s_j)$ for any schedule s . For any positive number q , define

$$\begin{aligned} \psi_q(j, r) &= \frac{\lceil q \cdot \psi(j, r) \rceil}{q} && \text{for any } j \in J \text{ and } r \in R; \\ \Phi_q(s) &= \sum_{j \in J} \psi_q(j, s_j) && \text{for any schedule } s \text{ for } J. \end{aligned}$$

Clearly, $q \cdot \psi_q(j, r)$ is an integer and $\psi(j, r) \leq \psi_q(j, r) \leq \psi(j, r) + \frac{1}{q}$ holds for each $j \in J$ and $r \in R$. Therefore,

$$\Phi(s) \leq \Phi_q(s) \leq \Phi(s) + \frac{|J|}{q} \quad (2)$$

holds for any schedule s of J . In other words, $\Phi_q(s)$ is the ‘‘rounded-up’’ energy consumption, which can be a good estimate for $\Phi(s)$ as long as q is sufficiently large. Finding s^* is NP-hard, but a feasible schedule s^q for J with minimum $\Phi_q(s^q)$ can be computed via the standard technique of dynamic programming as follows.

For any index $j \in J$ and any nonnegative k , let $\tau(j, k) = \infty$ signify that $\Phi_q(s) > k$ holds for any feasible schedule s for the job subset $\{1, 2, \dots, j\}$. If $\tau(j, k) \neq \infty$, let $\tau(j, k)$ be the minimum *completion* time required by any feasible schedule s for the job subset $\{1, 2, \dots, j\}$ with $\Phi_q(s) \leq k/q$. For notational brevity, define

$$\tau(0, k) = \begin{cases} 0 & \text{if } k \geq 0; \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

for any integer k . By the optimality of the earliest-deadline-first execution sequence, it is not difficult to verify that the following recurrence relation holds for any $j \in J$ and any positive integer k :

$$\tau(j, k) = \min_{r \in R} \begin{cases} \max(\tau(j-1, k - q \cdot \psi_q(j, r)), a_j) & \text{if } \max(\tau(j-1, k - q \cdot \psi_q(j, r)), a_j) \\ \quad + c_j/r & \quad + c_j/r \leq d_j; \\ \infty & \text{otherwise.} \end{cases} \quad (4)$$

Let k_q be the minimum k with $\tau(|J|, k) < \infty$. Clearly, $\Phi_q(s) = k_q/q$. Since each $q \cdot \psi_q(j, r)$ is an integer, a feasible schedule s^q for J with minimum $\Phi_q(s^q)$ can be obtained by a standard dynamic-programming algorithm, based upon Equations (3) and (4), in

$$O(|R||J| \cdot k_q) = O(|R||J| \cdot \Phi_q(s^q) \cdot q) \quad (5)$$

time and space. By Equation (2) and the optimality of s^* (respectively, s_q) with respect to Φ (respectively, Φ_q), we have

$$\Phi(s^*) \leq \Phi(s^q) \leq \Phi_q(s^q) \leq \Phi_q(s^*) \leq \Phi(s^*) + \frac{|J|}{q}. \quad (6)$$

It remains to determine q . Clearly, we want q to be sufficiently large so that $\Phi(s^q)$ can be close enough to $\Phi(s^*)$. For example, by $\epsilon \leq 1$, we can prove that

$$q \geq \frac{2|J|}{\epsilon \cdot \Phi_q(s^q)} \quad (7)$$

implies $\Phi(s^q) \leq (1 + \epsilon) \cdot \Phi(s^*)$ as follows. By Equation (7) and the last inequality in Equation (6), we have

$$\frac{|J|}{q} \leq \frac{\epsilon \cdot \Phi_q(s^q)}{2} \leq \frac{\epsilon \cdot \Phi(s^*)}{2} + \frac{\epsilon \cdot |J|}{2q}.$$

It follows that

$$\frac{|J|}{q} \leq \frac{\epsilon \cdot \Phi(s^*)}{2 - \epsilon} \leq \epsilon \cdot \Phi(s^*),$$

which by Equation (6) implies $\Phi(s^q) \leq (1 + \epsilon) \cdot \Phi(s^*)$.

Of course the immediate problem is that schedule s^q depends on the value of q . That is, we need to know the value of q in order to compute s^q , so it seems difficult to enforce $q \geq \frac{2|J|}{\epsilon \cdot \Phi_q(s^q)}$ at one shot. Fortunately, we can use the following trick of “doubling the value of q in each iteration”: Initially, we let q be

$$q' = \frac{|J|}{\epsilon \cdot \Phi(\hat{s})}. \quad (8)$$

In each iteration, we first compute s^q . If Equation (7) holds, we output the current s^q as a $(1 + \epsilon)$ -optimal solution. Otherwise, we double the value of q and then proceed to the next iteration. Let q'' be the value of q in the last iteration.

What is the required running time? By Equations (5) and (6) we know that the first iteration runs in $O(|J|^2|R|/\epsilon)$ time and space. Therefore, we focus on the case that the above procedure runs for more than one iteration. By Equations (5) and (6), we know that each iteration runs in $O(|R||J|(q \cdot \Phi(s^*) + |J|))$ time and space. Since the value of q is doubled in each iteration, the overall running time is

$$O\left(|R||J|\left(q'' \cdot \Phi(s^*) + |J| \log \frac{q''}{q'}\right)\right).$$

Since Equation (7) does not hold in the second-to-last iteration, we have

$$\frac{q''}{2} < \frac{2|J|}{\epsilon \cdot \Phi_{q''/2}(s^{q''/2})}. \quad (9)$$

Besides, by Equation (6), we know

$$\Phi(s^*) \leq \Phi_{q''/2}(s^{q''/2}). \quad (10)$$

Combining Equations (6), (9), and (10), we know $q'' \cdot \Phi(s^*) = O(|J|/\epsilon)$. By Equations (9) and (10) we have

$$q'' < \frac{4|J|}{\epsilon \cdot \Phi(s^*)}. \quad (11)$$

Combining the given relation of $\Phi(s^*)$ and $\Phi(\hat{s})$, Equations (8) and (11), we have

$$\log_2 \frac{q''}{q'} < \log_2 \frac{4\Phi(\hat{s})}{\Phi(s^*)} = O(\log \gamma).$$

The theorem is proved. \square

Since $\log \gamma$ is polynomial in the number of bits required to encode the input r and $\phi(r)$ for all $r \in R$, Lemma 4 provides a fully polynomial-time approximation scheme for the POWER-SAVING SCHEDULING problem when we are given a schedule \hat{s} satisfying $\Phi(s^*) \leq \Phi(\hat{s}) \leq \Phi(s^*) \cdot \gamma$ and the earliest-deadline-first sequence is known to be optimal, if there are no precedence constraints on J . Let $X_{R'}$ be the (imaginary) device with $R' = \{r \mid r_1 \leq r \leq r_{|R|}\}$ for any energy consumption function ϕ' that coincides with ϕ at all speeds in R . We need the following lemma to derive \hat{s} .

Lemma 5. *If preemption is allowed for the POWER-SAVING SCHEDULING problem, then there exists a polynomial-time algorithm to derive an optimal schedule on $X_{R'}$.*

Proof. Let s' be the schedule obtained by applying Lemma 1. That is, job j is to be executed at speed s'_j according to schedule s' . We now transform s' into \bar{s} so that \bar{s} is optimal and feasible on $X_{R'}$. We define J_l as $\{j \mid s'_j \geq r_1\}$. For each job $j \in J_l$, \bar{s} just copies the schedule of j on s' , including the speed setting and processing time intervals. For each job $j \in J - J_l$, if s' executes j in the interval $[z_1, z_2]$ (there could be more than one interval), then \bar{s} executes j in the interval $[z_1, z_1 + \frac{s'_j(z_2 - z_1)}{r_1}]$. It is clear that only one job in \bar{s} is processed at one time. Therefore, \bar{s} is feasible. We adopt the terminologies used in [35] to prove the optimality of \bar{s} : $g(I)$ for a time interval $I = [z, z']$ is defined as $g(I) = \frac{\sum_{j \in R_I} c_j}{z' - z}$, where R_I is the set of jobs satisfying $a_j \geq z$ and $d_j \leq z'$. A critical interval I^* satisfies $g(I^*) \geq g(I)$ for any interval I . Theorem 1 in [35] shows that there exists an optimal schedule S , which executes every job in R_{I^*} at the speed $g(I^*)$ completely within I^* and executes no other jobs during I^* . The algorithm for achieving Lemma 1 is obtained by computing a sequence of critical intervals iteratively. Therefore, it is clear that if the input job set J is feasible on $X_{R'}$, $g(I^*)$ must be no larger than $r_{|R|}$. We know that \bar{s} is a feasible schedule on $X_{R'}$ (allow preemption). Besides, the optimality of Theorem 1 in [35] fails on $X_{R'}$ only when $g(I^*) < r_1$. However, executing jobs at the speed r_1 results in an optimal solution in this situation. Therefore, \bar{s} is optimal on $X_{R'}$ and obtainable in $O(|J| \log^2 |J|)$ time if preemption is allowed. \square

Theorem 2. *The POWER-SAVING SCHEDULING problem admits a fully polynomial-time approximation scheme when (a) all jobs share a common arrival time, and (b) $\Pi = \emptyset$ and for any $j, j' \in J$, $a_j \leq a_{j'}$ implies $d_j \leq d_{j'}$.*

Proof. We first consider only the timing constraints of J in this paragraph. Based on Lemma 4, we just have to show that we can derive a schedule \hat{s} satisfying $\Phi(s^*) \leq \Phi(\hat{s}) \leq \Phi(s^*) \cdot \gamma$ efficiently and prove the optimality of the earliest-deadline-first execution sequence for these job sets. The *preemptive earliest deadline first rule* is defined in [18]: If a job arrives or is completed at time t , execute an arrived (ready) job with the earliest deadline at time t . Given a feasible schedule s , it is also feasible to schedule jobs according to the preemptive earliest-deadline-first rule by setting the processing speed of job j as

s_j [18] (if preemption is allowed). Since the job sets under considerations satisfy the condition $a_j \leq a_{j'}$ and $d_j \leq d_{j'}$ if $j' > j$, it is clear that each job in the resulting schedule following the preemptive earliest-deadline-first rule is non-preempted. The energy consumption for the resulting schedule remains, because s_j does not be increased or decreased. Therefore, the earliest-deadline-first sequence is known to be optimal for the POWER-SAVING SCHEDULING problem. Let \bar{s} be the resulting schedule from Lemma 5. Let s' be the schedule for J defined by letting s'_j be the smallest $r \in R$ with $\bar{s}_j \leq r$ for each $j \in J$ and schedule jobs according to the earliest-deadline-first execution sequence. It is clear that $\Phi(\bar{s}) \leq \Phi(s^*) \leq \Phi(s') \leq \Phi(\bar{s}) \cdot \gamma \leq \Phi(s^*) \cdot \gamma$. If $\Pi = \emptyset$, the fully polynomial-time approximation scheme is obtained by setting $\hat{s} = s'$.

In the following, we shall show how to deal with Π ($a_j = 0$ for all $j \in J$). Given an earliest-deadline-first feasible schedule s for J , we claim that we can transform s into another schedule s'' in $O(|J|^2)$ time such that $\Phi(s'') = \Phi(s)$ and s'' satisfies the precedence and timing constraints. If this claim stands, we can transform the derived schedule s^q in Lemma 4 (respectively, s' and s^* in the previous paragraph) into a feasible schedule without increasing $\Phi(s^q)$ (respectively, $\Phi(s')$ and $\Phi(s^*)$). This implies the existence of a fully polynomial-time approximation scheme. Initially, we let s'' be s . We consider a job j from job 2 to $|J|$. While considering job j , we look backward if there is a successor k to j is executed before j in s'' such that the ordered jobs J_{kj} executed between k and j are not successors to j . If such a job k exists, the execution sequence is modified by delaying k to be executed immediately after j finishes. Because all jobs are ready at time 0, job j and jobs in J_{jk} complete earlier. Since k is a successor to j ($d_k \geq d_j$) and all jobs are ready at time 0, the resulting schedule is feasible while considering jobs j , k , and J_{kj} . We repeat the previous procedure until no such a job k exists, and let s'' be the final schedule. It is clear that s'' is feasible for the job sets $\{1, 2, \dots, j\}$ after we perform the above rescheduling by considering job j . After we consider job $|J|$, s'' satisfies the precedence and timing constraints. The time complexity for the above rescheduling is $O(|J|^2)$. \square

4 Conclusion

This paper targets non-preemptive scheduling for minimization of energy consumption on devices that allow weakly dynamic voltage scaling. The problem is shown to be NP-hard even if the device has only two speeds and all jobs share the same arrival time and deadline. Moreover, we provide a fully polynomial-time approximation scheme of the NP-hard problem for two cases: (a) all jobs share a common arrival time, and (b) $\Pi = \emptyset$ and for any $j, j' \in J$, $a_j \leq a_{j'}$ implies $d_j \leq d_{j'}$.

An interesting direction for future research is to extend our approximation scheme to handle the overheads on voltage/speed switches.

References

1. A. Chandrakasan, S. Sheng, and R. Broderon. Lower-power CMOS digital design. *IEEE Journal of Solid-State Circuit*, 27(4):473–484, 1992.
2. L.-P. Chang, T.-W. Kuo, and S.-W. Lo. A dynamic-voltage-adjustment mechanism in reducing the power consumption of flash memory for portable devices. In *Proceedings of IEEE International Conference on Consumer Electronics*, pages 218–219, 2001.
3. 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. ACM Press, 2004.
4. J. Y. Chen, W. B. Jone, J. S. Wang, H.-I. Lu, and T. F. Chen. Segmented bus design for low-power systems. *IEEE Transactions on VLSI Systems*, 7(1):25–29, 1999.
5. Z. Chen, Q. Lu, and G. Tang. Single machine scheduling with discretely controllable processing times. *Operations Research Letters*, 21(2):69–76, 1997.
6. P. De, J. E. Dunne, J. B. Ghosh, and C. E. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research*, 45(2):302–306, 1997.
7. V. G. Deĭneko and G. J. Woeginger. Hardness of approximation of the discrete time-cost tradeoff problem. *Operations Research Letters*, 29(5):207–210, 2001.
8. M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Co., 1979.
9. V. Gutnik and A. P. Chandrakasan. Embedded power supply for low-power DSP. *IEEE Transactions on VLSI Systems*, 5(4):425–435, 1997.
10. I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava. Power optimization of variable voltage core-based systems. In *Proceedings of the 35th Annual Conference on Design Automation Conference*, pages 176–181. ACM Press, 1998.
11. O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *Journal of the ACM*, 22(4):463–468, 1975.
12. Intel Corporation. *28F016S5 5-Volt FlashFile Flash Memory Datasheet*, 1999.
13. S. Irani, S. Shukla, and R. Gupta. Competitive analysis of dynamic power management strategies for systems with multiple saving states. In *Proceedings of the Design Automation and Test Europe Conference*, 2002.
14. 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. Society for Industrial and Applied Mathematics, 2003.
15. 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.
16. W.-B. Jone, J. S. Wang, H.-I. Lu, I. P. Hsu, and J.-Y. Chen. Design theory and implementation for low-power segmented bus systems. *ACM Transactions on Design Automation of Electronic Systems*, 8(1):38–54, 2003.
17. S. Lee and T. Sakurai. Run-time voltage hopping for low-power real-time systems. In *Proceedings of the 37th Conference on Design Automation*, pages 806–809. ACM Press, 2000.
18. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.
19. A. Manzak and C. Chakrabarti. Variable voltage task scheduling algorithms for minimizing energy. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 279–282. ACM Press, 2001.
20. A. Manzak and C. Chakrabarti. Energy-conscious, deterministic I/O device scheduling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7):847–858, 2003.

21. A. Manzak and C. Chakrabarti. Variable voltage task scheduling algorithms for minimizing energy/power. *IEEE Transactions on VLSI Systems*, 11(2):270–276, 2003.
22. M. Pedram and J. M. Rabaey. *Power Aware Design Methodologies*. Kluwer Academic Publishers, 2002.
23. T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proceedings of the 1998 International Symposium on Low Power Electronics and Design*, pages 76–81. ACM Press, 1998.
24. T. Pering, T. Burd, and R. Brodersen. Voltage scheduling in the iparm microprocessor system. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 96–101. ACM Press, 2000.
25. J. Pouwelse, K. Langendoen, and H. Sips. Energy priority scheduling for variable voltage processors. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 28–33. ACM Press, 2001.
26. G. Quan and X. Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *Proceedings of the 38th Conference on Design Automation*, pages 828–833. ACM Press, 2001.
27. V. Raghunathan, M. B. Srivastava, and R. K. Gupta. A survey of techniques for energy efficient on-chip communication. In *Proceedings of the 40th Conference on Design Automation*, pages 900–905. ACM Press, 2003.
28. D. Shin and J. Kim. A profile-based energy-efficient intra-task voltage scheduling algorithm for real-time applications. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 271–274. ACM Press, 2001.
29. D. Shin, J. Kim, and S. Lee. Low-energy intra-task voltage scheduling using static timing analysis. In *Proceedings of the 38th Conference on Design Automation*, pages 438–443. ACM Press, 2001.
30. Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference*, pages 134–139. ACM Press, 1999.
31. Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design*, pages 365–368. IEEE Press, 2000.
32. M. Skutella. Approximation algorithms for the discrete time-cost tradeoff problem. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 501–508. Society for Industrial and Applied Mathematics, 1997.
33. M. Skutella. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, 23(4):909–929, 1998.
34. M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proceedings of Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.
35. 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.