Optimal Speed Scaling Algorithms under Speed Change Constraints

Zhi Zhang, Fei Li, Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: {zzhang8, lifei, aydin}@cs.gmu.edu

***Abstract*—In this paper, we investigate energy-aware real-time scheduling algorithms with speed change constraints. A processor is equipped with variable *clock frequency* (*speedy*) feature and is used to schedule a set of given jobs with deadlines. Each speed change involves time/energy overhead and recent studies show that it also impacts negatively the processor's *lifetime reliability*. Motivated by this, we study theoretical energy-aware scheduling problems with consideration of number and cost of speed changes. We associate a cost with each speed change to reflect its negative impact on the processor's lifetime reliability. We design speed schedules to satisfy all jobs' deadlines and optimize the energy consumption and the total cost incurred due to speed changes. Four related problems based on this framework are considered. We develop algorithms that perform arbitrarily close to the optimal and we also analyze their time complexities.**

***Keywords*-speed scaling algorithms; real-time scheduling; energy management; dynamic voltage scaling**

I. Introduction

Energy management remains an important problem for computer systems, and in particular, for real-time embedded systems. In this paper, we study energy-aware real-time job scheduling algorithms. We target on designing and analyzing algorithms for maximizing energy-usage efficiency with the consideration of system performance requirements.

The rapid advance of processor design technologies provides faster processors. Modern processors, such as those supported by *Intel SpeedStep* and *AMD PowerNOW!* technologies, have been equipped with a feature to vary the clock frequencies dynamically. The operating system is able to adjust the processor's *clock frequency* (*speed*) on the fly along with the supply voltage to execute jobs and reduce energy consumption at lower speeds. This functionality is called *speed scaling* and also *dynamic voltage scaling* [1]. Speed scaling is expected to satisfy some quality-of-service measures as well as to reduce overall energy cost, by manipulating modern processors' multiple speeds. In a seminal paper [2], Yao et al. initiated the algorithmic study of speed scaling techniques to schedule a given set of jobs with time constraints. Under the speed $f(t)$ at time t, the processor consumes energy $e(f(t))$ per time unit and the function $e(\cdot)$ is assumed to be convex. The objective is to construct a schedule satisfying all the jobs' deadline constraints and to minimize the total energy consumption, which is defined as $\int e(f(t))dt$. The algorithm in [2] has a running time of $O(n^3)$, where n is the number of jobs to be scheduled. Li et al. [3] improved the result to $O(n^2 \log n)$. Many settings and metrics based on this framework are studied in the literature, such as maximizing throughput [4], or minimizing the sum of energy consumption and (weighted) flow time of jobs [5], [6].

Existing studies considered energy minimization through speed scaling without much attention to the impact of speed changes. Such changes typically involve time and energy overhead. Moreover, recent studies indicate that the *lifetime reliability* of a CMOS circuit is directly related to the number and span of speed changes. For example, in [7], it is reported that *(hardware) failures, such as cracks and fatigue failures, are created not by sustained high temperatures, but rather by the repeated heating and cooling of sections of the processor. This phenomenon is referred to as 'thermal cycling'*. Thermal cycling is caused by the large difference in thermal expansion coefficients of metallic and dielectric materials, and leads to cracks and other permanent failures. Using MTTF (Mean-Time-To-Failure) to describe the expected processor's life, the following Coffin-Manson formula [8] is used to characterize a processor's lifetime reliability:

$$\text{MTTF} \propto \frac{1}{C_o(\Delta T_{mp} - \Delta T_o)^q x}, \tag{1}$$

where C_o is a material dependent constant, ΔT_{mp} is the entire temperature cycle-range of the device, ΔT_o is the portion of the temperature range in the elastic region, q is the Coffin-Manson exponent, and x is the frequency (number of occurrences per unit time) of thermal cycles [8]. Typically, $\Delta T_o \ll \Delta T_{mp}$ and $6 \leq q \leq 9$ for silicon materials. Simplifying Equation (1), we have

$$\text{MTTF} \propto \frac{1}{C_o \cdot \Delta T_{mp}^q \cdot x}. \tag{2}$$

Equation (2) clearly indicates that an algorithm which frequently changes the processor's speeds results in large x and ΔT_{mp}. Thus, such a schedule may introduce a large temperature cycle-range and therefore affect the processor's reliability adversely. Simulations in [7] have confirmed that various speed-scaling energy-aware policies have different

Table I
SUMMARY OF THE RESULTS IN THIS PAPER. (ALL THE ALGORITHMS RUN IN POLYNOMIAL TIME.)

algorithms	objectives to minimize	constraints
Algorithm 1	sum of energy consumption and costs of speed changes	-
Algorithm 2	energy consumption	bounded number of speed changes
Algorithm 3	number of speed changes	bounded energy consumption
Algorithm 4	energy consumption	bounded span of frequencies used

impacts on processor's reliability in terms of MTTF. The number of speed changes (x in Equation (2)) is a critical factor in determining a processor's reliability (MTTF in Equation (2)) under the thermal cycling phenomena.

A. Contributions

In this paper, we incorporate speed change constraints into speed-scaling scheduling algorithms. Limiting the number of speed changes is important for emerging reliability objectives in real-time embedded systems. The main purpose of this paper is to undertake a theoretical investigation of speed scaling algorithms by considering the costs and number of speed changes. Our contribution can be seen as the extension of the Yao et al.'s classical speed scaling algorithm [2] to incorporate the number and costs of speed changes. We consider four optimization problems and develop their corresponding solutions. We also analyze the running time complexities. The results in this paper are summarized in Table I.

We note that our results remain valid for arbitrary convex energy consumption functions $e(\cdot)$ with $e(0) = 0$. We do not require that $e(\cdot)$ should be in some single closed function form; it may be given by various closed formulas in different frequency ranges. A recent study considered the energy minimization for settings where the frequency of the bus and the memory can be adjusted independently, but without the speed change constraints [9]. In this paper, we assume that only the CPU's clock frequency can be adjusted. Extending our results to the settings of [9] is left as a future work.

B. Paper organization

In Section II, we formulate the problem of optimizing energy consumption and cost/number of speed changes into four combinatorial optimization problems. In Section III, we provide convex-programming-based algorithmic solutions to these problems and analyze their complexity. Related work is introduced in Section IV and conclusive remarks are given in Section V.

II. MODELS AND PROBLEM DEFINITION

We consider a single-processor setting. The processor has variable *clock frequencies* (*speeds*). Under a speed f, the processor consumes energy $e(f)$ per time unit and we simply assume that the function $e(\cdot)$ is convex and $e(0) = 0$. This setting is a generalization of the power model used in Yao's paper [2] and its successors [3], [10].

We note that in scaling speeds, the processor's frequency and supply voltage are both adjusted (dynamic voltage/frequency scaling). Hence, in the rest of the paper, we will understand that frequency/speed change always involves the corresponding voltage change [2].

We consider scheduling a set of n real-time jobs $\mathbf{J} = \{J_1,\ J_2,\ \ldots,\ J_n\}$. Each job J_i has a *release time* $r_i \in \mathbb{R}^+$, a *processing time* (also called *worst-case execution time*) $p_i \in \mathbb{R}^+$ and a *deadline* $d_i \in \mathbb{R}^+$. Under the speed f, it takes time p_i/f to complete the job J_i. We consider preemptive scheduling and assume that the cost of preemption is negligible.

The objective in this study is to design scheduling algorithms to finish all the jobs before their deadlines by considering the objectives of minimizing the energy consumption, as well as the *number* and *cost* of speed changes.

Definition 1 (Speed Schedule): A *speed schedule* can be viewed as a piece-wise constant curve, specifying the speed the processor employs in each time interval. Assume that the CPU speed changes m times during the execution. Then the speed schedule Ψ is defined by m intervals and the speed used in each of these intervals. Let the m time intervals be:

$$I_1 := (t_0,\ t_1],\ I_2 := (t_1,\ t_2],\ \ldots,\ I_m := (t_{m-1},\ t_m].$$

The triple $(t_{i-1},\ t_i,\ s_i)$ corresponds to the i^{th} time interval $I_i = (t_{i-1},\ t_i]$ $(0 < i \leq m$ and $t_0 = 0)$ in which the processor runs at a speed $s_i \geq 0$. Hence, the speed scheduler Ψ is specified by m triples:

$$\Psi:\ \{(t_0,\ t_1,\ s_1),\ (t_1,\ t_2,\ s_2),\ \ldots,\ (t_{m-1},\ t_m,\ s_m)\}.$$

Figure 1 illustrates an example schedule. The schedule employs 4 distinct speeds $f_1,\ f_2,\ f_3,\ f_4$ in 6 time intervals, where $s_1 = s_4 = f_1$, $s_2 = s_6 = f_4$, $s_3 = f_2$, and $s_5 = f_3$.

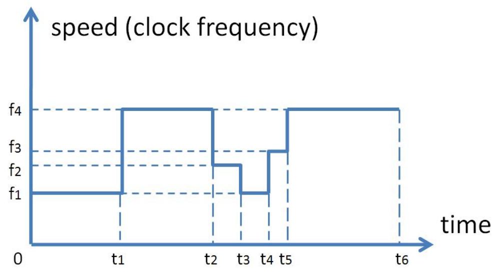

Figure 1. A piecewise curve describing the time intervals and the processor's speeds in each interval

We call time t_i a *speed switching point*. Without loss of generality, we assume that the processor is in *idle* state initially at time 0 ($s_0 = 0$) and gets back to the idle state after processing all the jobs ($s_{m+1} = 0$). Thus, a schedule with m time intervals have $m + 1$ speed switching points $t_0,\ t_1,\ \ldots,\ t_m$.

The total energy consumption of such a schedule is calculated as:

$$E^{\Psi} = \sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}).$$

For the example in Figure 1, we have

$$\begin{aligned} E^{\Psi} &= e(f_1)(t_1 - 0) + e(f_4)(t_2 - t_1) + e(f_2)(t_3 - t_2) \\ &\quad + e(f_1)(t_4 - t_3) + e(f_3)(t_5 - t_4) + e(f_4)(t_6 - t_5) \\ &= e(f_1)(t_1 + t_4 - t_3) + e(f_4)(t_2 - t_1 + t_6 - t_5) \\ &\quad + e(f_2)(t_3 - t_2) + e(f_3)(t_5 - t_4). \end{aligned}$$

To incorporate the penalty of changing clock frequencies, we consider that each speed change from the frequency s_i (in interval I_i) to the frequency s_{i+1} (in interval I_{i+1}) involves a *cost* $c_{i,i+1} \in \mathbb{R}^+$ ($c_{i,i+1}$ may, for example, reflect the speed change's negative impact on the processor's lifetime reliability). In [11], analysis and simulation results show the exponential (s^{α} with $\alpha > 2$) and super-linear ($s^{1+\epsilon}$) dependencies of the power on voltage (speed) s and temperature.

Along with the fact that the processor's reliability is a convex function of the temperature change (see Equation (2)), we assume that the cost of a speed change is a convex function of the difference between the previous and the new speed values. For instance, switching from f_4 to f_1 may be more costly than switching from f_4 to f_3, if $f_1 < f_3 < f_4$. Consequently, the function $c(\cdot)$ is convex and $c_{i,i+1}$ is the value of $c(\cdot)$ for $|s_i - s_{i+1}|$.

$$c_{i,i+1} := c(|s_i - s_{i+1}|), \text{ where } s_0 = s_{m+1} = 0. \quad (3)$$

We now present the formulation of four optimization problems.

$\mathcal{P}_1$. [Minimizing sum of energy consumption and costs incurred due to speed changes.]
Let E^{Ψ} denote the total energy consumed by the schedule Ψ to complete a set of jobs $\mathbf{J}$ by their deadlines. Assume Ψ has m time intervals. The total cost associated with all the clock speed changes during this schedule is $\sum_{i=0}^{m} c_{i,i+1}$ where s_0 and s_{m+1} are defined as 0 (see Equation (3)). In this problem, we seek to minimize $E^{\Psi} + \beta \sum_{i=0}^{m} c_{i,i+1}$, where β is a given constant. After normalizing $c_{i,i+1}$, we can remove β and formulate our problem as

$$\min.\left(\sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}) + \sum_{i=0}^{m} c'_{i,i+1}\right),$$

where $s_0 = s_{m+1} = 0$ and $c'_{i,i+1} = c_{i,i+1}/\beta$. (Note that $c'(\cdot) = c(\cdot)/\beta$ is still a convex function.) We can rename $c'_{i,i+1}$ as $c_{i,i+1}$.

$\mathcal{P}_2$. [Under a fixed number of clock speed changes.]
Let E^{Ψ} and m denote the total energy consumed and the total number of speed changes in the schedule Ψ to complete the jobs in $\mathbf{J}$ by their deadlines, respectively. Let M be the upper bound on the *number* of speed changes. The objective is to minimize E^{Ψ} subject to $m \leq M$. That is,

$$\min.\sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}), \quad \text{subject to } m \leq M.$$

The problem $\mathcal{P}_2$ considers the *number* of speed changes as a constraint.

$\mathcal{P}_3$. [Under a fixed energy budget.]
Let E^* denote the optimal (minimum) energy consumption required to complete all the jobs in $\mathbf{J}$ by their deadlines. (We do not have to know E^* beforehand.) Let E^b denote the energy budget that we are given. In this problem setting, we always have $E^b \geq E^*$. Let the schedule Ψ have m time intervals. The objective is to minimize the total number of speed changes m during the schedule subject to the constraint that the total energy consumption E^{Ψ} is bounded by E^b. That is,

$$\min.\ m, \quad \text{subject to } \sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}) \leq E^b.$$

$\mathcal{P}_4$. [Under a bound of span of frequencies used.]
Let E^{Ψ} and m denote the total energy consumed and the total number of time intervals of an schedule Ψ to complete the jobs $\mathbf{J}$ by their deadlines, respectively. Let $s^{\max} = \max_{1\leq i\leq m}\{s_i\}$ and $s^{\min} = \min_{1\leq i\leq m}\{s_i\}$, with $s^{\max} \geq s^{\min} \geq 0$. In other words $s^{\max}$ and $s^{\min}$ are the maximum and minimum speed used in Ψ while executing the jobs. The difference $s^{\max} - s^{\min}$ is defined as the *span* of the frequencies used in Ψ.
Let Q be the given upper bound on the span of the speeds that we do not want to exceed. The objective is to minimize E^{Ψ} subject to $s^{\max} - s^{\min} \leq Q$. That is,

$$\min.\sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}), \quad \text{subject to } s^{\max} - s^{\min} \leq Q.$$

For the problem $\mathcal{P}_4$, we bear the understanding that running the processor with a smaller speed span (small $s^{\max} - s^{\min}$ difference) results in less fluctuation of temperature (reduced ΔT_{mp} in Equation (2)), and thus, improves the chip's lifetime reliability (improved MTTF in Equation (2)).

In the following, we present convex-programming-based algorithmic solutions for the problems $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$. We analyze their performance as well. Note that Yao et al. [2] and Li et al. [3] presented algorithms to minimize the total energy consumption. The algorithms in [2] and [3] have no restrictions over the number of processor's speed changes. The models that we discussed above have their own algorithmic challenges. As we shall see, our solutions are different from [2], [3] that had the objective of minimizing energy alone. The problem $\mathcal{P}_2$ generalizes the well-studied Yao et al.'s model in [2], when we set the upper bound of speed changes M to a very large number.

III. Algorithms and Analysis

In this section, we provide algorithmic solutions for the problems $\mathcal{P}_1 - \mathcal{P}_4$.

A. Minimizing the sum of energy consumption and cost incurred due to speed changes

Assume the schedule Ψ has m time intervals $I_i = (t_{i-1},\ t_i]$ $(1 \le i \le m)$ and within each interval I_i, the processor keeps running at constant speed $s_i \ge 0$. The system does not consume any energy after finishing the last job. The objective is

$$\min.\left(\sum_{i=1}^{m} e(s_i)(t_i - t_{i-1}) + \sum_{i=0}^{m} c_{i,i+1}\right), \qquad (4)$$

where $c_{i,i+1}$ is scaled by a factor β from its definition in Equation (3).

Define OPT as an optimal algorithm minimizing the sum of energy consumption and the costs incurred due to speed changes (Equation (4)). Our job is to determine all the candidate values that t_i in Equation (4) can take. We note that the function $c(\cdot)$ is assumed to be a general convex function, hence determining the optimal schedule's speed switching points heavily depends on the function $c(\cdot)$ itself. In our technical report [12], we show that for an arbitrary convex cost function $c(\cdot)$, it is possible that the processor will need to change its speed in each time slot to optimize Equation (1). Instead of designing algorithms for some specific functions $c(\cdot)$, we study a large class of algorithms called *event-driven DVS (dynamic voltage scaling) algorithms*. The purpose of introducing an event-driven DVS algorithm for the problem $\mathcal{P}_1$ is to show that there exists an optimal convex-programming-based solution, and this solution's framework can be proved to generate optimal solutions for the other three problems $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$.

Definition 2 (Event-Driven DVS Algorithm): For *event-driven DVS algorithms*, speed changes (speed switching points) only happen at jobs' release times and/or deadlines.

We note here that event-driven DVS algorithms have the distinct advantage of keeping the run-time overhead due to DVS low, as opposed to DVS algorithms that require speed change at arbitrary points during execution: The CPU scheduler, that is invoked at task release times and deadlines, can also regulate the frequency according to the pre-determined speed schedule during the same invocation. As a result, we believe that our result regarding the optimality of event-driven DVS algorithms will prove very useful in practice.

Let $\mathbf{J} = \{J_1,\ J_2,\ \ldots,\ J_n\}$ denote the n jobs to be scheduled. A job J_j is represented by a triple $(r_j,\ d_j,\ p_j)$. Let $R = \{r_1,\ r_2,\ \ldots,\ r_n\}$ and $D = \{d_1,\ d_2,\ \ldots,\ d_n\}$. We use $Z = R \cup D$ to denote the union of all the release time and deadlines of jobs. Note $|Z| = |R \cup D| \le |R| + |D| \le 2n$. We sort all the values in Z in increasing order and index them as $z_1,\ z_2,\ \ldots,\ z_{n'}$ where $n' \le 2n$. Without loss of generality, we assume $z_1 = 0$. Before the last deadline $z_{n'}$, the time range is divided into $n' - 1$ non-overlapping intervals $(z_i,\ z_{i+1}]$, $\forall\ 1 \le i \le n' - 1$. We name the interval $T_{i,i'} := (z_i,\ z_{i'}]$ $(i' \ge i+1)$ as a *scheduling interval*. There are *at most* $\binom{n'}{2} = \frac{n'(n'-1)}{2} = O(n^2)$ such scheduling intervals. For each scheduling interval $T_{i,i'}$, we can compute its corresponding *workload* denoted by a variable $P_{i,i'}$ and *processing capacity* denoted by a variable $W_{i,i'}$, given the speed s'_l assumed for each interval $(z_{l-1},\ z_l]$.

$$P_{i,i'} = \frac{\sum_j p_j}{z_{i'} - z_i}, \quad \text{where } z_i < r_j < d_j \le z_{i'}. \qquad (5)$$

$$W_{i,i'} = \sum_{l=i+1}^{i'} s'_l,\ i < l \le i', \qquad (6)$$

where s'_l is the *speed variable* to denote at which speed the processor runs in the interval $(z_{l-1},\ z_l]$ $(s'_1 = s'_{|Z|+1} = 0)$. In order to complete all the jobs by their deadlines, the processing capacity should be at least the workload requirement for each time interval.

The remaining task is to determine s'_l such that all the jobs in $\mathbf{J}$ can be finished by their deadlines and the objective $\sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1}) + \sum_{l=2}^{|Z|+1} c(|s'_l - s'_{l-1}|)$ is minimized. We formulate this problem using a convex program CP_1 as below:

$$\begin{aligned} \min. \quad & \sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1}) + \sum_{l=2}^{|Z|+1} c(|s'_l - s'_{l-1}|) \\ \text{subject to} \quad & W_{i,i'} \ge P_{i,i'},\ \forall\ 1 \le i < i' \le |Z| \\ & s'_l \ge 0. \end{aligned}$$

In the following, we provide the analysis of correctness and time complexity for Algorithm 1, that computes an event-driven DVS schedule for the problem $\mathcal{P}_1$.

For a given set of real-time jobs with known processing times, preemptive EDF is optimal in the sense that any feasible job set can be also scheduled in a feasible manner by the EDF policy [13]. We have:

Lemma 1 (EDF Optimality): An optimal preemptive algorithm can have its jobs executed in a *canonical (deadline) order*, that is, for any two or more jobs ready to be executed for a given time, the job with the earliest deadline has the highest priority, with ties broken arbitrarily.

Algorithm 1 CONVEX-PROGRAMMING-BASED SOLUTION

```
Input: A job set $\mathbf{J} := \{J_1, J_2, \ldots, J_n\}$ where $J_j$ denoted by $\{r_j, d_j, p_j\}$
Output: A piecewise curve describing the time intervals and the speeds at which the processor runs in these intervals
1: Let $R = \{r_1, r_2, \ldots, r_n\}$, $D = \{d_1, d_2, \ldots, d_n\}$, and $Z = R \cup D$.
2: Sort $Z$ in increasing order. Let the distinct values be $z_i$, $1 \le i \le |Z|$. (Without loss of generality, assume $Z$ has exactly $|Z|$ distinct elements and $z_1 = 0$.)
3: Define $T_{i,i'} = (z_i, z_{i'}]$, for any $1 \le i < i' \le |Z|$. Define $s'_l$ for each $(z_{l-1}, z_l]$. Set $s'_1 = 0$ and $s'_{|Z|+1} = 0$.
4: for each pair $(i, i')$ with $1 \le i < i' \le |Z|$ do
5:     calculate
           $P_{i,i'} = \frac{\sum_j p_j}{z_{i'} - z_i}, \ z_i < r_j < d_j \le z_{i'}.$
6:     define
           $W_{i,i'} = \sum_{l=i+1}^{i'} s'_l, \ i < l \le i'.$
7: end for
8: Solve the convex program $CP_1$:
       min.        $\sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1}) + \sum_{l=2}^{|Z|+1} c(|s'_l - s'_{l-1}|)$
       subject to  $W_{i,i'} \ge P_{i,i'}, \ \forall\ 1 \le i < i' \le |Z|$
                   $s'_l \ge 0.$
9: return a schedule running jobs in canonical order using speeds $\{s'_l\}$.
```

Lemma 2 (Convexity of CP_1): The formulation CP_1 in Algorithm 1 is a convex program.

Proof: We study the objective first. Note that all the z-values are chosen from $Z = R \cup D$. Thus, $e(s'_l)(z_l - z_{l-1})$ is a non-negative linear combination of the convex function $e(s'_l)$, which is convex as well. The norm $|s'_l - s'_{l-1}|$ of an affine function $s'_l - s'_{l-1}$ is a convex function. Note that all the variables $|s'_l - s'_{l-1}|$ are non-negative, so, by the definition of function $c(\cdot)$, $c(|s'_l - s'_{l-1}|)$ is a non-decreasing function. As the cost function $c(\cdot)$ is assumed convex, the composition $c(|s'_l - s'_{l-1}|)$ preserves convexity.

We then study the constraints. All the constraints are convex ones (actually, linear ones). (Note that all the calculated values $P_{i,i'}$ are constants but not variables in our formulation CP_1. We can rewrite $W_{i,i'} \ge P_{i,i'}$ as $W_{i,i'} - P_{i,i'} - \delta_{i,i'} = 0$, where $\delta_{i,i'} \ge 0$.) ■

Let $G(\cdot)$ denote the running time of evaluating $e(\cdot)$ and $c(\cdot)$ and their first and second derivatives for all the constraints in the convex program CP_1.

Theorem 1 (Optimal Event-Driven Schedule for $\mathcal{P}_1$): Algorithm 1 generates an optimal event-driven speed schedule and has a running time of $O(\max\{n^4,\ n \cdot G\})$, where n is the number of jobs to be scheduled and G is the time of evaluating $e(\cdot)$ and $c(\cdot)$ and their first and second derivatives for all the constraints.

Proof: The correctness of Theorem 1 depends on Lemma 1 and Lemma 2. First, Lemma 1 guarantees that running jobs in the canonical manner does not hurt the optimality. Second, we claim that $W_{i,i'} \ge P_{i,i'}$ is the necessary condition to ensure a feasible schedule; this has been proved in [14]. Combining these two observations and Lemma 2, we conclude that Algorithm 1 generates an optimal event-driven schedule. Note that the convex program in Algorithm 1 can be solved using the interior-point method [15] with the solutions for variables s'_l bounded by arbitrarily small errors.

Now, we analyze the running complexity. Note $|Z| \le 2n$ and thus, $|Z| = O(n)$. Sorting Z takes time $O(n \log n)$. The number of scheduling intervals $T_{i,i'}$ is $O(n^2)$. There are $O(n^2)$ variables for $P_{i,i'}$ and $W_{i,i'}$. Calculating $P_{i,i'}$ takes time $O(n^2)$. (A straightforward way of calculating $P_{i,i'}$ takes time $O(n^3)$. Here is one alternative way with faster running time: We map each pair (r_j, d_j) of a job J_j to each corresponding time in Z and result a convex bipartite graph. Working on this convex bipartite graph improves the calculation time of getting $P_{i,i'}$ to be $O(n^2)$.) Before we get to solve the convex program CP_1, we pay time $O(n^2)$ for the preliminary work. Note that CP_1 has $O(n^2)$ constraints. We use the interior-point method [15] to solve CP_1 optimally (arbitrarily close to optimal). Thus, it takes time $O(\max\{n^3,\ n^2 \cdot O(n^2),\ n \cdot G(e(\cdot),\ c(\cdot))\}) = O(\max\{n^4,\ n \cdot G(e(\cdot),\ c(\cdot))\})$, where $G(\cdot)$ is the time of evaluating $e(\cdot)$ and $c(\cdot)$ and their first and second derivatives for all the constraints [16]. ■

B. Minimizing energy consumption under limited number of speed changes

In this section, we consider speed schedules under a limited number of speed changes. Let M be the upper bound of the number of speed changes that a speed schedule Ψ is allowed to schedule jobs. We study the problem $\mathcal{P}_2$: minimizing the total energy consumption subject to satisfying all the jobs' deadline constraints and bounding the number of clock speed changes by M.

We now provide a lemma that guarantees that the number of constraints in our convex program is polynomial in the number of jobs in $\mathbf{J}$ – the same lemma ensures also that an optimal schedule for $\mathcal{P}_2$ can be an event-driven DVS schedule.

Lemma 3 (Upper Bound of # of Scheduling Intervals): An optimal algorithm OPT has its speed changes only happen at time points in the set $Z = R \cup D$.

Proof: We prove Lemma 3 by using the contradiction method. Without loss of generality, assume that OPT has

M line-segments to indicate the time intervals in which the processor executes jobs at different speeds. Let t be the first (earliest) time such that t is a speed switching point and t is neither a release time nor a deadline of a job. Thus, at time $t - \epsilon$, either the processor is *idle* (that is, no job is being executed) or some job, say J_j, has not been finished and it is being executed at this point.

1) Assume the processor is idle at time $t - \epsilon$. At time t, a new job must be released; since otherwise, in order to save more energy without violating the bound of number of speed changes, the processor can keep the same speed until the next speed change (if the next speed at time t is larger than the one at time $t - \epsilon$) or the processor can immediately slow down its speed at the time when the processor finished the last job before time $t - \epsilon$ (if the next speed at time t is smaller than this one at time $t - \epsilon$). This contradiction shows that t must be some release time.
2) Assume the processor is executing some job J_j at time $t - \epsilon$ and $t \neq d_j$. From Lemma 1, we know that in this OPT, all the jobs are executed in a canonical order. Particularly, if no job is released at time t, then the processor should keep the same speed or fasten (respectively, slow down) its speed before t in order to reduce the total energy consumption. Thus, we have that either at time t some job is released or all the jobs have been completed by time t.

Based on the above discussion, we claim that Lemma 3 holds, due to the convexity of the energy consumption function $e(\cdot)$ with $e(0) = 0$. ∎

In the following, we design algorithms for the problem $\mathcal{P}_2$. Similar to the case of $\mathcal{P}_1$, we sort all the values in Z in increasing order and index them as $z_1,\ z_2,\ \ldots,\ z_{n'}$ where $n' \leq 2n$. Thus, the whole time is divided into $n' - 1$ non-overlapping intervals $(z_i,\ z_{i+1}]$ in which the processor runs at possible positive speeds, $\forall 1 \leq i \leq n' - 1$. The interval $T_{i,i'} := (z_i,\ z_{i'}]$ ($i' \geq i + 1$) is a scheduling interval. There are *at most* $\binom{n'}{2} = \frac{n'(n'-1)}{2} = O(n^2)$ such scheduling intervals. For each scheduling interval $T_{i,i'}$, we calculate its corresponding workload $P_{i,i'}$ and processing capacity $W_{i,i'}$ as those in Equation (5) and Equation (6), given the speed s'_l assumed for each interval $(z_{l-1},\ z_l]$.

The remaining task is to determine s'_l such that all the jobs in $\mathbf{J}$ can be finished by their deadlines and the objective is to bound the number of speed changes by M. Note that the following piece-wise function is a convex one.

$$c_{i,i+1} = c(|s_i - s_{i+1}|) = \begin{cases} \epsilon, & \text{if } |s'_i - s'_{i+1}| < \delta, \\ H, & \text{otherwise.} \end{cases}$$

where H is a large positive number, and ϵ is a small positive constant. $c(\cdot)$ is convex since it can be represented by $c(x) = \max\{y_1(x),\ y_2(x)\}$ for $x \in \mathbb{R}^+$, where $y_1(x) = \epsilon$, $\forall x \geq 0$, $y_2(x) = H$, $\forall x \geq \delta$ and $y_2(x) = 0$, $\forall 0 \leq x < \delta$.

By using the above function $c(\cdot)$, we set the objective to minimizing the energy consumption $\sum_{l=1}^{|Z|} e(s'_l)(z_l - z_{l-1})$ subject to the total cost incurred due to speed changes bounded by the sum of $M \cdot H$ and the costs associated with those intervals running at similar frequencies (where the frequency difference is bounded by a specified input value δ). To ensure that the final schedule has no more than M speed changes, we need to set $H \gg \epsilon \cdot |Z|$.

We present the algorithm (Algorithm 2) for this problem using the convex programming technique.

Algorithm 2 BOUNDED # OF SPEED CHANGES (M)

```
Input: A job set $\mathbf{J} := \{J_1,\ J_2,\ \ldots,\ J_n\}$ and $J_j = \{r_j,\ d_j,\ p_j\}$. The upper bound of number of speed changes $M$
Output: A piecewise curve describing the time intervals and the speeds at which the processor runs in these intervals
1: Let $R = \{r_1,\ r_2,\ \ldots,\ r_n\}$, $D = \{d_1,\ d_2,\ \ldots,\ d_n\}$, and $Z = R \cup D$.
2: Sort $Z$ in increasing order. Let the distinct values be $z_i$, $1 \leq i \leq |Z|$. (Without loss of generality, assume $Z$ has exactly $|Z|$ distinct elements and $z_1 = 0$.)
3: Define $T_{i,i'} = (z_i,\ z_{i'}]$, for any $1 \leq i < i' \leq |Z|$. Define $s'_l$ for each $(z_{l-1},\ z_l]$. Set $s'_1 = 0$ and $s'_{|Z|+1} = 0$.
4: for each pair $(i,\ i')$ with $1 \leq i < i' \leq |Z|$ do
5:     calculate
           $$P_{i,i'} = \frac{\sum_j p_j}{z_{i'} - z_i},\ z_i < r_j < d_j \leq z_{i'}.$$
6:     define
           $$W_{i,i'} = \sum_{l=i+1}^{i'} s'_l,\ i < l \leq i'.$$
7: end for
8: Solve the convex program $CP_2$:
       $$\begin{aligned} \min. \quad & \sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1}) \\ \text{subject to} \quad & W_{i,i'} \geq P_{i,i'},\ \forall\ 1 \leq i < i' \leq |Z| \\ & \sum_{l=2}^{|Z|+1} c(|s'_l - s'_{l-1}|) \leq M \cdot H + \epsilon \cdot |Z| \\ & s'_l \geq 0. \end{aligned}$$
9: return a schedule running jobs in canonical order using speeds $\{s'_l\}$.
```

Theorem 2 (Optimal Schedule for $\mathcal{P}_2$): Algorithm 2 generates a schedule arbitrarily close to the optimal and has a running time of $O(\max\{n^4,\ n \cdot G\})$, where G is the time of evaluating $e(\cdot)$ and its first and second derivatives for all the constraints.

Proof: It is obvious that $C\mathcal{P}_2$ in Algorithm 2 is a convex program and it can be solved to obtain the optimal values for variables s'_l (up to arbitrarily small errors). In the following, we illustrate that Algorithm 2 solves our problem $\mathcal{P}_2$ correctly and we analyze its running complexity.

Lemma 3 guarantees that for $\mathcal{P}_2$, we only need to consider a speed schedule's speed switching points from the time points in Z. Similar to the analysis of $\mathcal{P}_1$, we conclude that Algorithm 2 generates an optimal solution. The running complexity of Algorithm 2 is similar to that of Algorithm 1. The details can be found in our technical report [12]. ■

C. Minimizing number of speed changes subject to bounded energy consumption

In this section, we study the problem of minimizing the number of speed changes subject to satisfying all jobs' deadline constraints and energy consumption bounded by a given budget. This problem is motivated by budgeting energy consumption during execution in energy-constrained environments.

Let E^* denote the optimal (minimum) energy consumption required to satisfy all the jobs' time constraints in $\mathbf{J}$. Let E^b denote the energy budget that we are given. In our problem setting for $\mathcal{P}_3$, we always have $E^b \geq E^*$. The objective is to minimize the total number of clock speed changes m subject to keeping the total consumed energy E^Ψ bounded by E^b. That is,

$$\min.\ m, \quad \text{subject to } \sum_{i=1}^{m} e(s_i)(z_i - z_{i-1}) \leq E^b.$$

Actually, $\mathcal{P}_3$ is the dual problem of $\mathcal{P}_2$. We can use $\mathcal{P}_2$'s objective as our $\mathcal{P}_3$'s constraint and $\mathcal{P}_2$'s constraint $\sum_{l=1}^{|Z|} c(s'_l,\ s'_{l-1}) \leq M \cdot H + \epsilon|Z|$ as $\mathcal{P}_3$'s objective. We still use the function

$$c_{i,i+1} = c(|s_i - s_{i+1}|) = \begin{cases} \epsilon, & \text{if } |s'_i - s'_{i+1}| < \delta, \\ H, & \text{otherwise.} \end{cases}$$

where H is a large positive number, and ϵ is a small positive constant. We immediately have the following algorithm.

Theorem 3 (Optimal Schedule for $\mathcal{P}_3$): Algorithm 3 generates a schedule close to optimal arbitrarily and has a running time of $O(\max\{n^4,\ n \cdot G\})$, where G is the time of evaluating $e(\cdot)$ and its first and second derivatives for all the constraints.

Proof: Note that Lemma 3 holds for the problem $\mathcal{P}_3$ as well. The proof of Theorem 3 is almost the same as the one for $\mathcal{P}_2$ (see Theorem 2). We skip the details. ■

D. Minimizing energy consumption subject to bounded span of frequencies

In this section, we study the problem of minimizing the energy consumption subject to the span of frequencies used in the schedule bounded by a given number. Let E^Ψ and m denote the total energy consumed and the total number of speed changes by the schedule Ψ to complete the jobs in $\mathbf{J}$ by their deadlines. Let Q be a given upper bound of the span of the speeds used by Ψ. Let $s^{\max} = \max_{1\leq i\leq m}\{s_i\}$ and $s^{\min} = \min_{1\leq i\leq m}\{s_i\}$. The objective is to minimize E^Ψ subject to $s^{\max} - s^{\min} \leq Q$. We note here that $\mathcal{P}_4$ generalizes the problem $\mathcal{P}_2$, given Q is allowed to be sufficiently large.

Let us consider the following idea for $\mathcal{P}_4$: Assume we have the min-energy schedule for a given job set $\mathbf{J}$. (This min-energy schedule can be achieved by the algorithms in [2], [3] in time $O(n^2 \log n)$.) We then realize that the processor has to run at a speed $s^{\max}$ during an interval $[z,\ z']$; otherwise, some job belonging to this interval cannot be finished by its deadline. This speed $s^{\max}$ ensures the highest speed to guarantee the feasibility of finishing $\mathbf{J}$.

Algorithm 3 BOUNDED ENERGY BUDGET (E^b)

```
Input: A job set $\mathbf{J} := \{J_1,\ J_2,\ \ldots,\ J_n\}$ and $J_j = \{r_j,\ d_j,\ p_j\}$. The energy budget $E^b$
Output: A piecewise curve describing the time intervals and the speeds at which the processor runs in these intervals
1: Let $R = \{r_1,\ r_2,\ \ldots,\ r_n\}$, $D = \{d_1,\ d_2,\ \ldots,\ d_n\}$, and $Z = R \cup D$.
2: Sort $Z$ in increasing order. Let the distinct values be $z_i$, $1 \leq i \leq |Z|$. (Without loss of generality, assume $Z$ has exactly $|Z|$ distinct elements and $z_1 = 0$.)
3: Define $T_{i,i'} = (z_i,\ z_{i'}]$, for any $1 \leq i < i' \leq |Z|$. Define $s'_l$ for each $(z_{l-1},\ z_l]$. Set $s'_1 = 0$ and $s'_{|Z|+1} = 0$.
4: for each pair $(i,\ i')$ do
5:    calculate
          $P_{i,i'} = \frac{\sum_j p_j}{z_{i'} - z_i},\ z_i < r_j < d_j \leq z_{i'}.$
6:    define
          $W_{i,i'} = \sum_{l=i+1}^{i'} s'_l,\ i < l \leq i'.$
7: end for
8: Solve the convex program $CP_3$:
          min.        $\sum_{l=2}^{|Z|+1} c(s'_l,\ s'_{l-1})$
          subject to  $W_{i,i'} \geq P_{i,i'},\ \forall\ 1 \leq i < i' \leq |Z|$
                      $\sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1}) \leq E^b$
                      $s'_l \geq 0.$
9: return a schedule running jobs in canonical order using speeds $\{s'_l\}$.
```

According to $\mathcal{P}_4$, all the speeds of a schedule that we are going to design should be in the range of $[s^{\max} - Q,\ s^{\max}]$. This range indicates that the processor should keep running at speed $\geq s^{\max} - Q$ all along the schedule before all the jobs are finished. We thus proceed as if we have "two virtual processors" for the single variable-speed processor: One (called A), which is running at speed $s^{\max} - Q$, and the other one (called B) which is running with variable speeds within range $[0,\ Q]$.

Based on above ideas, to solve $\mathcal{P}_4$, we partition the job set into two parts: some parts of jobs that can be finished by running A with constant speed $s^{\max} - Q$, and some parts of jobs can be finished by the variable speed processor B. For these two processors, we apply two algorithms.

1) First, we run EDF (earliest-deadline-first policy) over all the jobs using speed $s^{\max} - Q$, assuming $s^{\max} - Q > 0$. (If $s^{\max} \leq Q$, we simply use Yao et al.'s algorithm [2], [3] as our solution.) If a job J_j cannot be finished by its deadline, we simply cut the unfinished processing time part at its deadline d_j and name it J'_j with remaining processing time p'_j.
2) Second, for those unfinished processing time parts of jobs, we run an optimal algorithm to find out min-energy schedule. This step can be solved using a convex program.

The idea of partitioning the jobs into finished part and unfinished part using speed $s^{\max} - Q$ provides us the feasibility of calculating the schedule's speed switching points using a convex program. Algorithm 4 shows the details of our algorithm. To save space, we skip the analysis, which is similar to that of Theorem 2.

IV. Related Work

The first theoretical energy-efficient job scheduling model is studied by Yao et al. [2]. In this model, jobs have release times and deadlines, and a continuous spectrum of speeds is available. This framework is by far the most extensively studied algorithmic speed scaling problem. A straightforward implementation has running time of $O(n^3)$. Li et al. [3] improved this result, giving a $O(n^2 \log n)$-time algorithm, and this is by far the best algorithm. Li et al. [3] and Kwon and Kim [17] also studied the discrete setting in which the processor has k discrete speeds. Kwon and Kim [17] achieved a $O(n^3)$-time algorithm. Li et al. [3] got a $O(k \cdot n \log n)$ algorithm in minimizing the total consumed energy. In above min-energy models, the number or cost of frequency changes is unrestricted. Our work studies min-energy speed schedulers with considerations of number and cost frequency changes and it is a natural next step in this line of research.

V. Conclusions

Motivated by enhancing processor's lifetime reliability from the perspective of designing speed-scaling algorithms, we investigate energy-aware scheduling algorithms in this paper. Our contributions include a few scheduling algorithms for one model and three variants, optimizing energy consumption and number/cost of frequency changes. We apply the convex programming techniques for the general model. Based on this framework, we develop three polynomial-time optimal solutions for three important variants. The algorithms that we provide are proved arbitrarily close to optimal.

In our future research, we will study the relationship between the frequency and the temperature/heat generated by the processor in order to get a better understanding processor's lifetime reliability. Then we can provide a more precise model on processor's lifetime reliability and algorithmic solutions for it.

Algorithm 4 Bounded Span of Frequencies Used (Q)

```
Input: A job set $\mathbf{J} := \{J_1,\ J_2,\ \ldots,\ J_n\}$ and $J_j = \{r_j,\ d_j,\ p_j\}$. The bounded span of frequencies used $Q$
Output: A piecewise curve describing the time intervals and the speeds at which the processor runs in these intervals
1: Let $R = \{r_1,\ r_2,\ \ldots,\ r_n\}$, $D = \{d_1,\ d_2,\ \ldots,\ d_n\}$, and $Z = R \cup D$.
2: Sort $Z$ in increasing order. Let the distinct values be $z_i$, $1 \leq i \leq |Z|$. (Without loss of generality, assume $Z$ has exactly $|Z|$ distinct elements.)
3: Define $T_{i,i'} = (z_i,\ z_{i'}]$, for any $1 \leq i < i' \leq |Z|$. Define $s'_l$ for each $(z_{l-1},\ z_l]$. Set $s'_1 = 0$ and $s'_{|Z|+1} = 0$.
4: Calculate the min-energy $E^*$ and the maximum speed $s^{\max}$ that $E^*$ has.
5: if $s^{\max} \leq Q$ then
6:     return $E^*$ as the solution;
7: else
8:     simulate the the jobs in $\mathbf{J}$ over a machine with a constant speed $s^{\max} - Q$. For any unfinished job $J_j$, cut it by the deadline. Name it $J_{j'}$. $J_{j'}$ has a remaining processing time $p'_j$.
9:     solve the convex program $CP_4$:
           $\min.\ \sum_{l=2}^{|Z|} e(s'_l)(z_l - z_{l-1})$
           subject to $W_{i,i'} \geq P_{i,i'},\ \forall\ 1 \leq i < i' \leq |Z|$
           $0 \leq s'_l \leq Q.$
       {If $J_j$ is an unfinished job $J_{j'}$ after we finish step 8, then the remaining processing time $p'_j$ is used in calculating $P_{i,i'}$ as in Equation (5).}
10:    return a schedule running jobs in canonical order using speeds $\{s'_l + s^{\max} - Q\}$.
11: end if
```

Acknowledgment

This material is based upon work supported by the U.S. National Science Foundation under Grants No. CCF-0915681, CNS-1016855 and CNS-546244 (CAREER Award). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 53, no. 10, pp. 584–600, 2004.

[2] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995, pp. 374–382.

[3] M. Li, A. C. Yao, and F. F. Yao, "Discrete and continous min-energy schedules for variable voltage processors," *Proceedings of the National Academy of Sciences of the USA*, vol. 103, no. 11, pp. 3983–3987, 2005.

[4] H.-L. Chan, W.-T. Chan, T.-W. Lam, L.-K. Lee, K.-S. Mak, and P. W. H. Wong, "Energy efficient online deadline scheduling," in *Proceedings of the 18th Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007, pp. 795–804.

[5] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, p. Article No. 49, 2007.

[6] K. Pruhs, P. Uthaisombut, and G. Woeginger, "Getting the best response for your erg," *ACM Transactions on Algorithms (TALG)*, vol. 4, no. 3, p. Article No. 38, 2008.

[7] A. K. Coskun, R. Strong, D. M. Tullsen, and T. S. Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the 11th ACM International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, 2009, pp. 169–180.

[8] "Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C," http://www.jedec.org.

[9] H. Yun, P.-L. Wu, A. Arya, T. Abdelzaher, C. Kim, and L. Sha, "System-wide energy optimization for multiple DVS components and real-time tasks," in *Proceedings of the 22nd Euromicro Conference on Real-Time Systems (ECRTS)*, 2010, pp. 133–142.

[10] M. Li, B. J. Liu, and F. F. Yao, "Min-energy voltage allocation for tree-structured tasks," *Journal of Combinatorial Optimization*, vol. 11, no. 3, pp. 305–319, 2006.

[11] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage-estimation considering power supply and temperature variations," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED)*, 2003, pp. 78–83.

[12] Z. Zhang, F. Li, and H. Aydin, "Optimal speed scaling algorithms under speed change constraints," Department of Computer Science, George Mason University, http://cs.gmu.edu/ lifei/papers/speed_scheduling.pdf, Tech. Rep., 2011.

[13] M. Dertouzos, "Control robotics: the procedural control of physical processes," in *Proceedings of the IFIP Congress*, 1974, pp. 807–813.

[14] S. Baruah, "A general model for recurring real-time tasks," in *Proceedings of the 19th IEEE International Real-Time Systems Symposium (RTSS)*, 1998, pp. 114–122.

[15] Y. Nesterov and Nemirovskii, *Interior-Point Polynomial Methods in Convex Programming*. Society for Industrial and Applied Mathematics (SIAM), 1994.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[17] W.-C. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 4, no. 1, pp. 211–230, 2005.