Tradeoff between Energy and Throughput for Online Deadline Scheduling

Ho-Leung Chan Tak-Wah Lam Rongbin Li
Department of Computer Science, University of Hong Kong
Emails: {hlchan, twlam, rbli}@cs.hku.hk

Abstract

The past few years have witnessed a number of interesting online algorithms for deadline scheduling in the dynamic speed scaling model (in which a processor can vary its speed to manage its rate of energy usage). These algorithms aim at minimizing the energy usage to complete all given jobs or to achieve the maximum throughput. Note that throughput is the primary concern for these algorithms. In this paper we consider a more general objective for studying the tradeoff between throughput and energy for deadline scheduling. Specifically, we assume each job is associated with a user-defined value (or importance) and a deadline. We allow a scheduling algorithm to discard some of the jobs (i.e., not finishing them by their deadlines), and the objective is to minimize the total energy usage plus the total value of jobs discarded.

We give new online algorithms under both the unbounded-speed and bounded-speed models. When the maximum speed is unbounded, we give an $O(1)$-competitive algorithm. This algorithm relies on a key notion called the profitable speed, which is the maximum speed beyond which processing a job costs more energy than the value of the job. When the processor has a bounded maximum speed T, we show that no $O(1)$-competitive algorithm can exist and more precisely, the competitive ratio grows with the penalty ratio of the input, which is defined as the ratio between the maximum profitable speed of a job to the maximum speed T. On the positive side, we give an algorithm with a competitive ratio whose dependency on the penalty ratio almost matches the lower bound.

1 Introduction

Energy efficiency is a major concern not only for mobile devices, but also for large-scale server farms like those operated by Google [13]. Recently, it has been reported that the average energy cost for running a server exceeds the purchase cost of the server [9]. To improve energy efficiency, major chip manufacturers like Intel and AMD now produce processors equipped with a technology called *dynamic voltage scaling*. Specifically, it allows operating systems or application software to dynamically vary the processor speed so as to manage the energy usage. Running at low speed reduces energy usage drastically, yet we still want to maintain some kind of quality of service (QoS). These conflicting objectives have imposed new challenges to the research on scheduling. In this paper, the QoS concerned is the throughput, i.e., total size or value of jobs completed by their deadlines.

The history. The theoretical study of energy-efficient online scheduling was initiated by Yao, Demers and Shenker [14]. They considered online deadline scheduling on a processor that can vary its speed dynamically between $[0, \infty)$. When the processor runs at speed s, the rate of energy usage, denoted by $P(s)$, can be modeled as s^α, where $\alpha > 1$ is a constant commonly believed to be 2 or 3 (it is determined by the physical property of the hardware technology). Jobs with different sizes and deadlines arrive over time in an online fashion. Jobs are preemptive and a

preempted job can be resumed later at the point of preemption. The objective is to minimize the total energy usage subject to completing all jobs by their deadlines. [14] proposed two online algorithms AVR and OA, and showed that AVR is $(2^{\alpha-1}\alpha^{\alpha})$-competitive on minimizing the energy (if $\alpha = 3$, AVR is 108-competitive). After about a decade, Bansal, Kimbrel and Pruhs [7] showed that OA is indeed better and is α^{α}-competitive; they also gave another algorithm BKP that is $O(e^{\alpha})$-competitive (which is better than OA if α is large). Recently, Bansal et al. [6] showed that no algorithm can have a competitive ratio better than $e^{\alpha-1}/\alpha$, and they also gave an algorithm qOA that is $4^{\alpha}/(2\sqrt{e\alpha})$-competitive. When $\alpha = 3$, the competitive ratio of qOA can be fine tuned to 6.7.

All the above work assumes that the processor has unbounded maximum speed and can always complete every job on time. Chan et al. [10] extended the study of energy-efficient scheduling to a more realistic setting where a processor can only vary its speed between 0 to some fixed maximum speed T. Since the maximum speed is bounded, it is possible that no algorithm can complete all the given jobs. It is natural to consider the case where the optimal algorithm aims at maximizing the throughput (which is the total size of jobs completed by their deadlines), while minimizing the energy usage subject to this maximum throughput. They gave an online algorithm that is 14-competitive on throughput and $(\alpha^{\alpha} + 4^{\alpha}\alpha^2)$-competitive on energy. Later, Bansal et al. [4] gave an improved algorithm that is 4-competitive on throughput, while the competitive ratio of energy remains the same. This algorithm is optimal in terms of throughput since any algorithm is at least 4-competitive on throughput even if we ignore the energy concern [11].

Tradeoff between energy and throughput. To our knowledge, existing work on energy-efficient deadline scheduling all assumes that throughput is the primary concern. That means, a scheduling algorithm first aims at maximizing the throughput and then minimizes the energy usage subject to the maximum throughput. With the growing importance of energy saving, this assumption may not be valid and some systems may actually prefer to trade throughput for better energy efficiency. For example, imagine the following scenario. There is a web server whose users are divided into different levels of importance. During the peak period, it may be desirable to drop the requests from less important users if the extra energy used for speeding up the processor to serve these requests costs more than the revenue generated by these requests. Note that when the server load is low, requests from less important users could be served at low speed. The energy usage is much smaller and could make these jobs profitable.

Our results. To cater for the above situations, we initiate studying the tradeoff between throughput and energy. Specifically, we assume that each job is associated with a deadline and a user-defined *value*, the latter is about the importance of the job (e.g., the value can be the job size or simply a fixed constant). A scheduling algorithm may choose to finish only a subset of the given jobs by their deadlines and discard the rest. The objective is to minimize the total energy usage plus the total value of jobs discarded. The objective of minimizing the total energy usage and value discarded has the following interpretation. From an economic point of view, a user would estimate the cost for one unit of energy and the revenue generated for each job. By normalizing the cost for one unit of energy to be one and assigning the normalized revenue for each job as its value, minimizing the total energy usage plus value discarded is equivalent to maximizing the total profit of the system.

In this paper, we first study the tradeoff in the unbounded speed model. Notice that the problem of minimizing the total energy usage plus total value discarded is a generalization of the classical problem of minimizing the total energy usage for completing all jobs, thus inheriting

any lower bound result from the latter. The argument is as follows. Consider a set of jobs whose values are set to be sufficiently large, then the optimal offline algorithm and any competitive online algorithm will not discard any jobs, and the problem of minimizing energy plus value discarded is reduced to the problem of minimizing the energy usage subject to completing all jobs. Furthermore, since the value discarded is zero in this case, any c-competitive algorithm for the new objective gives a c-competitive algorithm for the classical objective. Recall that for the classical objective, no online algorithm has a competitive ratio better than $e^{\alpha-1}/\alpha$ [6]. This lower bound is also valid for the new objective of minimizing energy plus value discarded.

On the positive side, when the maximum speed is unbounded, we give an $O(1)$-competitive algorithm called PS. Precisely, the competitive ratio of PS is $\alpha^\alpha + 2e\alpha$. The main idea is about a notion called *profitable speed*, which is the maximum speed beyond which processing the job costs more energy than the value of the job. Roughly speaking, the algorithm works as follows. When a job is released, PS calculates the OA schedule for all admitted jobs together with the new job. The new job is admitted if the OA schedule processes the new job with a speed at most c times the profitable speed, where c is a carefully chosen constant; otherwise the new job is discarded immediately. Though PS might look simple, the analysis is non-trivial. We first upper bound the value discarded by PS in terms of the energy used by PS plus the energy and value discarded of the optimal offline algorithm. Then we bound the energy usage of PS using a potential analysis.

For the bounded speed model, we show that the new objective becomes more difficult by giving a non-constant lower bound on the competitive ratio of any online algorithm. In particular, we define the *penalty ratio* of an input instance as the ratio of the maximum profitable speed of a job to the maximum processor speed T. We show that the competitive ratio of any algorithm is $\Omega(\max\{e^{\alpha-1}/\alpha, \Gamma^{\alpha-2+1/\alpha}\})$, where Γ is the penalty ratio. The lower bound holds even if all jobs have the value equal to the size. On the other hand, we adapt the algorithm PS to the bounded-speed setting and show that its competitive ratio is $\alpha^\alpha + 2\Gamma^{\alpha-1}(\alpha+1)^{\alpha-1}$. Note that the dependency on the penalty ratio almost mathes the lower bound.

Remark on an alternative objective. Another and perhaps a more natural approach to studying the tradeoff between throughput and energy is to consider the objective of maximizing the total value of jobs completed by their deadlines minus the total energy usage. However, we first notice that this objective, unlike the one for minimizing the total energy usage plus value discarded, is no longer a generalization of the classical model of minimizing total energy subject to completing all jobs. That is, a c-competitive algorithm for this maximization objective no longer gives a c-competitive algorithm for the classical model. More importantly, even in the unbounded-speed setting with the restriction that job value equals job size, this maximization objective is intractable as we can easily construct an instance where any online algorithm has total throughput minus energy arbitrarily close to zero or even zero, while an offline algorithm can obtain at least a finite throughput minus energy. We consider optimizing the total energy plus value discarded to avoid the singularity issue of getting a zero or close to zero value in the objective function.

Other related work. Energy efficiency has attracted a lot attention from the scheduling community in the past few years, see, e.g., [1] for a survey. Besides the related work already mentioned, there is another well-studied problem with similar flavor as ours, which is about energy-efficient flow time scheduling. In that problem, jobs with arbitrary sizes, but with no deadlines, arrive over time. The flow time of a job is the length of the duration from it is released until it is completed. The objective is to complete all jobs and to minimize the total energy

usage plus the total flow time of the jobs. The objective defined in this paper is motivated in part by this energy-plus-flow-time objective. Albers and Fujiwara [2] were the first to study this energy plus flow time objective. Following a chain of works [5, 8, 12], Andrew et al. [3] have finally given an 2-competitive algorithm for minimizing energy plus flow time.

2 Preliminaries

We give the formal problem definition and review the algorithm OA used in our algorithms.

Problem definition. We consider online scheduling of jobs on a single processor. Each job j has a release time $r(j)$, size $p(j)$, deadline $d(j)$ and a value $v(j)$. We use J and $v(J)$ to denote a sequence of jobs and their total values. Jobs are preemptive and a preempted job can be resumed later at the point of preemption. The processor can run at any speed in $[0, \infty)$ in the unbounded speed model and can run at speed in $[0, T]$ in the bounded speed model, where T is a fixed constant. In any case, the rate of energy usage of the processor is s^α when it is running at speed s, where $\alpha > 1$ is also a constant.

Formally speaking, let $s(t)$ be the speed of the processor at time t. Then the total energy usage of the processor is $\int_0^\infty (s(t))^\alpha dt$. Let $s(j,t)$ denote the speed at which a job j is being processed at time t. The algorithms in this paper do not use time sharing; yet, if time sharing is allowed, we require that $\sum_j s(j,t) \leq s(t)$ at any time t. A job j is said to be completed by its deadline $d(j)$ if $\int_{r(j)}^{d(j)} s(j,t)dt \geq p(j)$; and j is said to be discarded otherwise.

The objective is to minimize the total energy usage plus the total value of jobs discarded. We denote Opt as the optimal offline schedule which minimizes the objective for any input J. An algorithm is said to be γ-competitive if for any input J, the total energy usage plus the total value discarded is at most γ times that of Opt.

Algorithm OA. Our algorithms will make use of the algorithm OA [7,14]. Below we review its definition and some of its properties. At any time t, OA defines a sequence of times $t_0, t_1, \ldots$ as follows. Let S be the jobs remaining at time t. Let $t_0 = t$. For $i = 1, 2, \ldots,$, let t_i be the latest time after t_{i-1} such that $\frac{w(t_{i-1},t_i)}{t_i - t_{i-1}}$ is maximized, where $w(t_{i-1}, t_i)$ is the total remaining size for jobs in S with deadline in $(t_{i-1}, t_i]$. The interval $I_i = (t_{i-1}, t_i]$ is called the *i-th critical interval*, and the quantity $\rho_i = \frac{w(t_{i-1},t_i)}{t_i - t_{i-1}}$ is called the *density* of I_i. OA processes the jobs in the order of EDF (earliest deadline first) and sets the speed during each critical interval $(t_{i-1}, t_i]$ to be ρ_i. Note that in the OA schedule as defined, the speed decreases from one critical interval to another. It has been shown that this OA schedule uses the minimum energy to complete S. If a new job j arrives after time t, the OA schedule will be recomputed then at time $r(j)$.

Property 1. *Consider an OA schedule and assume a job j arrives at time $r(j)$. Let S be the jobs remaining just before j arrives and let $OA(S)$ be the OA schedule just before j arrives. Let $OA(S \cup \{j\})$ be the re-calculated OA schedule just after j arrives. Then,*

(i) In $OA(S \cup \{j\})$, j is processed by a constant speed $s(j)$. Furthermore, the speed of $OA(S \cup \{j\})$ during the period $[r(j), d(j)]$ is at least $s(j)$.

(ii) Let I be any set of disjoint intervals after time $r(j)$. The total amount of work scheduled in I by $OA(S \cup \{j\})$ is at least the total amount of work scheduled in I by $OA(S)$, but at most the total amount of work scheduled in I by $OA(S)$ plus $p(j)$.

3 Unbounded Speed Model

This section considers the unbounded speed model where the processor can run at any speed in $[0, \infty)$. We present an algorithm PS(c), which stands for Profitable Speed with parameter c, and show that it is $(\alpha^\alpha + 2e\alpha)$-competitive when setting $c = \alpha^{(\alpha-2)/(\alpha-1)}$.

Before we present our algorithm, we define the notion of profitable speed. For any job j, let $u(j) = v(j)/p(j)$ be the *value density* of j.

Definition 1. *For any job j, the* profitable speed *of j, denoted $\tilde{s}(j)$, equals $(u(j))^{1/(\alpha-1)}$.*

Fact 1. *If we complete a job j at a constant speed equal to $\tilde{s}(j)$, then the energy usage on processing j equals the value of j.*

Proof. The energy usage equals $(\tilde{s}(j))^\alpha \frac{p(j)}{\tilde{s}(j)} = (\tilde{s}(j))^{\alpha-1} p(j) = u(j)p(j) = v(j)$. □

Intuitively, the profitable speed $\tilde{s}(j)$ is a "boundary speed" suggesting whether we should complete or discard j. If the speed needed to complete j is larger than $\tilde{s}(j)$, the energy usage on processing j will be larger than its value and discarding it (instead of completing it) is more beneficial. On the other hand, if the speed needed is smaller than $\tilde{s}(j)$, completing j is "profitable". Roughly speaking, our algorithm completes j only when it can be completed at speed at most $c \cdot \tilde{s}(j)$, where $c > 0$ is a constant.

3.1 Algorithm PS(c)

Let $c > 0$ be any parameter. The algorithm PS(c) maintains a list Q of admitted jobs, which is empty initially. When a job arrives, it is immediately admitted into Q or discarded. PS(c) only processes and completes jobs in Q. Details are as follows.

Algorithm PS(c)

- **Job execution.** At any time t, PS(c) uses OA to schedule the jobs in Q. (Note that in the literature, the OA schedule is defined and analyzed based on the entire input rather than a subset.)
- **Job admission.** When a job j arrives at time $r(j)$, let S be the set of jobs remaining in Q just before j arrives. PS(c) calculates the OA schedule for $S \cup \{j\}$. Let $s(j)$ be the speed of j in this OA schedule. PS(c) admits j into Q if $s(j) \leq c \cdot \tilde{s}(j)$; and j is discarded immediately otherwise.
- **Job Completion.** When a job j in Q is completed, remove it from Q.

By definition, OA always completes the jobs given to it no later than their respective deadlines. Thus, PS(c) also meets the deadline of every job in Q. The main result of this section is about the competitiveness of PS(c) on minimizing energy plus value discarded.

Theorem 1. *For any $c > 0$, PS(c) is $\left((1 + \frac{b^{\alpha-1}}{(cb-1)^\alpha}) \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\} + \max\{b^{\alpha-1}, 1\}\right)$-competitive on energy plus value discarded, for any $b > \frac{1}{c}$*

By choosing the parameter c to be $\alpha^{\frac{\alpha-2}{\alpha-1}}$ and considering $b = \frac{\alpha+1}{c} = \frac{\alpha+1}{\alpha^{(\alpha-2)/(\alpha-1)}}$, the competitive ratio becomes $\alpha^\alpha + 2\alpha(1 + \frac{1}{\alpha})^{\alpha-1}$. Since $(1 + \frac{1}{\alpha})^{\alpha-1} < e$, we obtain the following.

Corollary 2. *PS($\alpha^{\frac{\alpha-2}{\alpha-1}}$) is $(\alpha^\alpha + 2e\alpha)$-competitive for any $\alpha > 1$.*

To prove Theorem 1, we analyze the energy and the value discarded separately. Consider any input job sequence J and parameter $c > 0$. Let E_a and E_o be the total energy usage of PS(c) and Opt, respectively. Similarly, let D_a and D_o be the value discarded by PS(c) and Opt, respectively. We will prove the following two lemmas concerning the value discarded and energy usage of PS(c), whose proofs are given in the following subsections.

Lemma 3. *$D_a \le \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a + b^{\alpha-1} E_o + D_o$, for any $b > \frac{1}{c}$.*

Lemma 4. *$E_a \le \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}(E_o + D_o)$.*

Lemmas 3 and 4 together imply Theorem 1.

Proof of Theorem 1. For Opt, the total energy usage plus value discarded is $E_o + D_o$. The theorem follows from the following bound of $E_a + D_a$.

$$\begin{aligned} E_a + D_a &\le \left(1 + \tfrac{b^{\alpha-1}}{(cb-1)^\alpha}\right) E_a + b^{\alpha-1} E_o + D_o \\ &\le \left(1 + \tfrac{b^{\alpha-1}}{(cb-1)^\alpha}\right) \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}\Big(E_o + D_o\Big) + b^{\alpha-1} E_o + D_o \end{aligned}$$

□

3.2 Value discarded by PS(c)

This section analyzes D_a and proves Lemma 3. Let $J_D \subseteq J$ be the subset of jobs discarded by PS(c). We further divide J_D into J_{D1} and J_{D2}, which include the jobs that are completed and discarded by Opt, respectively. $D_a = v(J_{D1}) + v(J_{D2}) \le v(J_{D1}) + D_o$. To prove Lemma 3, it is sufficient to show that $v(J_{D1}) \le \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a + b^{\alpha-1} E_o$.

Let j be an arbitrary job in J_{D1}. Let $I(j)$ be the set of maximal time intervals during which Opt processes j. Denote $|I(j)|$ as the total length of the intervals in $I(j)$. Denote $E_a(I(j))$ and $E_o(I(j))$ as the energy usage by PS(c) and Opt during $I(j)$, respectively. We will bound $v(j)$ by $E_a(I(j))$ and $E_o(I(j))$. Intuitively, if $|I(j)|$ is small, Opt completes $p(j)$ units of work in a short period of time and $E_o(I(j))$ should be relatively large. On the other hand, if $|I(j)|$ is large, then $E_a(I(j))$ is relatively large since PS(c) discards j and PS(c) must run at relatively high speed during $I(j)$. Details are as follows.

Lemma 5. *Let j be any job in J_{D1}. Then $v(j) \le \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a(I(j)) + b^{\alpha-1} E_o(I(j))$ for any $b > \frac{1}{c}$.*

Proof. To ease the discussion, let us denote $\tilde{\ell}(j)$ as the time to complete j if at speed $\tilde{s}(j)$, i.e., $\tilde{\ell}(j) = p(j)/\tilde{s}(j)$. Note that $p(j) = \tilde{s}(j) \cdot \tilde{\ell}(j)$. Let $b_j = |I(j)|/\tilde{\ell}(j)$.

Note that Opt completes exactly $p(j)$ units of work in $I(j)$ and Opt runs at the speed $p(j)/|I(j)|$ throughout $I(j)$. Therefore,

$$E_o(I(j)) = \left(\frac{p(j)}{|I(j)|}\right)^\alpha |I(j)| = \left(\frac{\tilde{\ell}(j) \cdot \tilde{s}(j)}{|I(j)|}\right)^{\alpha-1} p(j) = \frac{u(j)}{b_j^{\alpha-1}} \cdot p(j) = \frac{v(j)}{b_j^{\alpha-1}} \tag{1}$$

where the last equality comes from the definition that $u(j) = v(j)/p(j)$.

Since j is discarded by PS(c), consider the time $r(j)$ when j arrives. Let S be the set of jobs remaining in Q just before j arrives. Let OA(S) and OA($S \cup \{j\}$) be the OA schedules starting from time $r(j)$ for S and $S \cup \{j\}$, respectively, assuming no other jobs arrive. Since j is discarded, the speed of j in OA($S \cup \{j\}$) is at least $c \cdot \tilde{s}(j)$. Since all intervals in $I(j)$ are

completely inside $[r(j), d(j)]$, by Property 1 (i), the speed of OA($S \cup \{j\}$) throughout these intervals is at least $c \cdot \tilde{s}(j)$. Hence, the total work done by OA($S \cup \{j\}$) during $I(j)$ is at least $c \cdot \tilde{s}(j) \cdot |I(j)|$. By Property 1 (ii), the work done by OA(S) in the intervals in $I(j)$ is at least $c \cdot \tilde{s}(j) \cdot |I(j)| - p(j)$. Again by Property 1 (ii), if some more jobs arrive after j, the amount of work scheduled to the intervals in $I(j)$ may only increases. Therefore, we have

$$\begin{aligned} E_a(I(j)) &\geq \left(\frac{c \cdot \tilde{s}(j) \cdot |I(j)| - p(j)}{|I(j)|}\right)^\alpha |I(j)| \\ &= \left(\frac{c \cdot \tilde{s}(j) \cdot b_j \tilde{\ell}(j) - \tilde{s}(j) \cdot \tilde{\ell}(j)}{b_j \tilde{\ell}(j)}\right)^\alpha b_j \cdot \tilde{\ell}(j) \\ &= \frac{(c \cdot b_j - 1)^\alpha}{b_j^\alpha} \cdot (\tilde{s}(j))^\alpha \cdot b_j \tilde{\ell}(j) = \frac{(cb_j - 1)^\alpha}{b_j^{\alpha-1}} u(j) \tilde{s}(j) \tilde{\ell}(j) = \frac{(cb_j - 1)^\alpha}{b_j^{\alpha-1}} v(j) \quad (2) \end{aligned}$$

Finally, for any $b > 1/c$, there are two cases. If $b > b_j$, then by (1), $v(j) = b_j^{\alpha-1} E_o(I(j)) < b^{\alpha-1} E_o(I(j))$. Otherwise, $b \leq b_j$, then by (2), $v(j) \leq \frac{b_j^{\alpha-1}}{(cb_j-1)^\alpha} E_a(I(j)) \leq \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a(I(j))$, where the last inequality comes from the fact that function $f(x) = \frac{x^{\alpha-1}}{(cx-1)^\alpha}$ is decreasing when $x > \frac{1}{c}$. Hence for all $b > \frac{1}{c}$, the lemma holds. □

Lemma 3 follows immediately then.

Proof of Lemma 3. Note that for any two jobs j and j' in J_{D1}, $I(j)$ and $I(j')$ are disjoint. Hence, by summing up the inequality in Lemma 5 over all jobs in J_{D1}, we have $v(J_{D1}) \leq \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a + b^{\alpha-1} E_o$, and Lemma 3 follows. □

3.3 Energy usage of PS(c)

This section analyzes E_a and proves Lemma 4. We will use a potential function, which is similar to the one used in analyzing OA [7]. However, a major difference in our problem is that both PS(c) and Opt may discard jobs, so the set of jobs scheduled by the two algorithms can be different. In particular, when a job j is admitted by PS(c) but discarded by Opt, our analysis needs to relate the extra energy usage of PS(c) on processing j to the value of j discarded by Opt. Intuitively, this extra energy can be bounded because PS(c) admits j only if its speed is at most c times the profitable speed. Details are as follows.

W.L.O.G., we assume that Opt admits a job j at $r(j)$ if Opt will complete j; otherwise, Opt discards j immediately. Let $E_a(t)$ and $E_o(t)$ be the energy usage of PS(c) and Opt, respectively, by time t. Let $D_o(t)$ be the total value of jobs discarded by Opt by time t. Let $s_a(t)$ and $s_o(t)$ be the speed of PS(c) and Opt, respectively, at time t. We will define a potential function $\Phi(t)$ satisfying the following conditions.

- *Boundary condition:* $\Phi(t) = 0$ before any job arrives and after all deadlines are passed.
- *Running condition:* At any time t without job arrival, $\frac{d}{dt} E_a(t) + \frac{d}{dt} \Phi(t) \leq \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\} \frac{d}{dt}(E_o(t) + D_o(t))$.
- *Arrival condition:* When a job j arrives at time t, let $\Delta\Phi(t)$ and $\Delta D_o(t)$ denote the change of $\Phi(t)$ and $D_o(t)$, respectively, due to the arrival of j. Then $\Delta\Phi(t) \leq \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\} \Delta D_o(t)$.

Similar to [7, 10], with such a potential function, we can prove by induction on time that

$$\forall t, \qquad E_a(t) + \Phi(t) \le \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}(E_o(t) + D_o(t))$$

which implies Lemma 4 as $\Phi(t) = 0$ after all deadlines are passed.

Definition of the potential function. Consider any time t. For any $t'' \ge t' \ge t$, let $w_a(t', t'')$ be the total remaining size for jobs in the admitted list Q of PS(c) with deadlines in $(t', t'']$. Recall that PS(c) processes Q by OA, which defines a sequence of times $t_0, t_1, t_2, \ldots$, where $t_0 = t$ and for $i = 1, 2, \ldots$, t_i is the latest time after t_{i-1} such that $\rho_i = \frac{w_a(t_{i-1}, t_i)}{t_i - t_{i-1}}$ is maximized. We call $I_i = (t_{i-1}, t_i]$ as the i-th critical interval. On the other hand, consider the schedule of Opt, and let $w_o(t', t'')$ be the total remaining size for jobs admitted by Opt by time t with deadlines in $(t', t'']$. The potential function $\Phi(t)$ is defined as

$$\Phi(t) = \alpha \sum_{i \ge 1} \rho_i^{\alpha-1}(w_a(t_{i-1}, t_i) - \alpha w_o(t_{i-1}, t_i)) \tag{3}$$

It is easy to see that $\Phi(t)$ satisfies the boundary condition. We prove that it satisfies the arrival and running conditions as follows. Unlike the previous potential analysis [7, 10], the arrival condition is non-trivial as PS(c) and Opt may have different decision on admitting a new job.

Arrival condition. When a job j arrives at time t, there are four cases depending on whether PS(c) and Opt admit j. We first consider the two easier cases where PS(c) discards j. Since PS(c) discards j, all critical intervals I_i's, their densities ρ_i's and $w_a(t_{i-1}, t_i)$'s do not change. Furthermore, $w_o(t_{i-1}, t_i)$ may only increases. Hence, $\Delta\Phi(t) \le 0$. On the other hand, $\Delta D_o(t) \ge 0$ depending on whether j is discarded by Opt, so the arrival condition holds.

The following discussion considers the case where PS(c) admits j. For simplicity, we first assume that $p(j)$ is small so that admitting j only affect the density of the critical interval that contains $d(j)$ while all other critical intervals are unaffected. Let I_k be the only interval affected and let ρ and ρ' be the density of I_k just before and after j is admitted, respectively. Let $w_a(k)$ and $w_o(k)$ denote the total remaining size for jobs in PS(c) and Opt, respectively, with deadlines in I_k just before j is admitted. Let $|I_k|$ denote $t_k - t_{k-1}$. Then, $\rho = \frac{w_a(k)}{|I_k|}$, and $\rho' = \frac{w_a(k)+p(j)}{|I_k|}$. We first bound $\Delta\Phi(t)$.

Lemma 6. *Let $\Delta\Phi(t)$ be the change in $\Phi(t)$ if j is admitted by PS(c) and discarded by Opt. Then $\Delta\Phi(t) \le \alpha^2 c^{\alpha-1} v(j)$.*

Proof. Note that $w_o(k)$ remains unchanged as Opt discards j. By definition,

$$\begin{aligned}
\Delta\Phi &= \alpha(\rho')^{\alpha-1}(w_a(k) + p(j) - \alpha w_o(k)) - \alpha\rho^{\alpha-1}(w_a(k) - \alpha w_o(k)) \\
&\le \alpha(\rho')^{\alpha-1}(w_a(k) + p(j)) - \alpha\rho^{\alpha-1} w_a(k) \\
&= \frac{\alpha}{|I_k|^{\alpha-1}}\left((w_a(k) + p(j))^\alpha - w_a(k)^\alpha\right)
\end{aligned}$$

Note that for any convex function $f(z)$ and any real numbers $y > x$, we have $f(y) - f(x) \le f'(y)(y - x)$, where f' denotes the derivative of f. Putting $f(z) = z^\alpha$ where $\alpha > 1$ and consider $y = w_a(k) + p(j)$ and $x = w_a(k)$, we have that

$$\frac{\alpha}{|I_k|^{\alpha-1}}\left((w_a(k) + p(j))^\alpha - w_a(k)^\alpha\right) \le \frac{\alpha}{|I_k|^{\alpha-1}}\alpha(w_A(k) + p(j))^{\alpha-1}p(j) = \alpha^2(\rho')^{\alpha-1}p(j)$$

Since j is admitted by PS(c), we have $\rho' \le c \cdot \tilde{s}(j)$ by definition. It follows that $\Delta\Phi \le \alpha^2 c^{\alpha-1}(\tilde{s}(j))^{\alpha-1}p(j) = \alpha^2 c^{\alpha-1}u(j)p(j) = \alpha^2 c^{\alpha-1}v(j)$ □

Therefore, if Opt discards j, $\Delta D_o = v(j)$, and hence $\Delta\Phi(t) \le \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}\Delta D_o$. Finally, if Opt admit j, $\Delta D_o = 0$. The analysis on $\Delta\Phi(t)$ is similar to that in [7]. Note that both $w_a(k)$ and $w_o(k)$ is increased by $p(j)$. Hence,

$$\begin{aligned}\Delta\Phi(t) &= \alpha(\rho')^{\alpha-1}\Big(w_a(k)+p(j)-\alpha(w_o(k)+p(j))\Big) - \alpha\rho^{\alpha-1}\Big(w_a(k)-\alpha w_o(k)\Big)\\ &= \frac{\alpha}{|I_k|^{\alpha-1}}\Big[((w_a(k)+p(j))^{\alpha-1}\Big(w_a(k)+p(j)-\alpha(w_o(k)+p(j))\Big)\\ &\quad -w_a(k)^{\alpha-1}\Big(w_a(k)-\alpha w_o(k)\Big)\Big]\end{aligned}$$

The last term is at most zero by setting $q = w_a(k)$, $r = w_o(k)$, $\delta = p(j)$ to Lemma 7. Hence, $\Delta\Phi(t) \le 0 = \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}\Delta D_o$.

Lemma 7. *([7]) Let $q, r, \delta \ge 0$ and $\alpha \ge 1$, then $(q+\delta)^{\alpha-1}(q+\delta-\alpha(r+\delta)) - q^{\alpha-1}(q-\alpha r) \le 0$.*

So far, we have assumed that $p(j)$ is small and only one critical interval is affected. To remove the assumption, we can follow the technique of [7, 10]. If $p(j)$ is large, we can split j into two jobs j_1 and j_2 so that their release times, deadlines and value densities are the same as j, and $p(j_1)$ is the smallest size such that some critical intervals merge or a critical interval splits into multiple ones. $p(j_2) = p(j) - p(j_1)$. Note that $\Phi(t)$ does not change due to merging or splitting of critical intervals. The above argument can show that the arrival condition holds after $p(j_1)$ is admitted. Furthermore, we can repeat the division recursively on j_2 to conclude that the arrival condition holds.

Running condition. Analysis for the running condition is similar to that in [7]. We include it here for completeness. Consider any time t without job arrival. Let $s_a(t)$ and $s_o(t)$ be the speed of PS(c) and Opt, respectively. Then $E_a(t)$ and $E_o(t)$ are increasing at the rates of $(s_a(t))^\alpha$ and $(s_o(t))^\alpha$, respectively, while $D_o(t)$ remains constant. Note that to prove the running condition, it is sufficient to prove that $(s_a(t))^\alpha + \frac{d}{dt}\Phi(t) - \alpha^\alpha(s_o(t))^\alpha \le 0$. In the following, we omit the parameter t for simplicity. E.g., we write s_a to mean $s_a(t)$.

PS(c) processes jobs by OA, which in turns processes jobs by EDF. Hence, at time t, PS(c) is processing a job with deadline in I_1. Furthermore, $s_a = \rho_1$ and $w_a(t_0, t_1)$ is decreasing at a rate of s_a. Suppose Opt is processing a job with deadline in I_k, where $k \ge 1$. Then $w_o(t_{k-1}, t_k)$ is decreasing at a rate of s_o. Therefore $\frac{d}{dt}\Phi = \alpha\rho_1^{\alpha-1}(-s_a) + \alpha^2\rho_k^{\alpha-1}s_o \le \alpha\rho_1^{\alpha-1}(-s_a) + \alpha^2\rho_1^{\alpha-1}s_o = -\alpha s_a^\alpha + \alpha^2 s_a^{\alpha-1}s_o$, where the inequality comes from $\rho_k \le \rho_1$. Finally, $s_a^\alpha + \frac{d}{dt}\Phi - \alpha^\alpha s_o^\alpha \le (1-\alpha)s_a^\alpha + \alpha^2 s_a^{\alpha-1}s_o - \alpha^\alpha s_o^\alpha$. By considering the last expression as a function of s_a where s_o is a constant, the last expression can be shown to be non-positive by simple differentiation.

4 Bounded Speed Model

This section considers the bounded speed model where the processor speed can be any value in $[0, T]$. T is called the maximum speed of the processor. We first define a quantity called the penalty ratio of a job sequence. As we will see, the difficulty of scheduling a job sequence depends on its penalty ratio.

Definition 2. *Consider scheduling in the bounded speed model with maximum speed T. The* penalty ratio *of a job, denoted $\Gamma(j)$, equals $\tilde{s}(j)/T$. The* penalty ratio *of a sequence J of jobs, denoted $\Gamma(J)$ or simply Γ if J is clear in context, equals the maximum penalty ratio of all jobs in J, i.e., $\Gamma = \max_{j\in J}\Gamma(j)$.*

4.1 Lower Bound

We first give a non-constant lower bound for the bounded speed model.

Theorem 8. *For the bounded speed model, the competitive ratio of any algorithm is at least* $\min\{\Gamma^{\alpha-2+1/\alpha}, \frac{1}{2}\Gamma^{\alpha-1}\}$*, where* Γ *is the penalty ratio of the input job sequence.*

Proof. Let Alg be any algorithm. The theorem is obviously true if $\Gamma \leq 1$. In the following, let $\Gamma > 1$ be the targeted penalty ratio. Let $x > 1$ be a variable to be set later. At time 0, release a job j_1 with $d(j_1) = x$, $p(j_1) = T$ and $v(j_1) = T^\alpha \Gamma^{\alpha-1}$. Note that $\tilde{s}(j_1) = (v(j_1)/p(j_1))^{1/(\alpha-1)} = T\Gamma$ and $\Gamma(j_1) = \tilde{s}(j_1)/T = \Gamma$. At time 1, one of the following two cases occurs.

- If Alg has completed j_1 by time 1, then Alg must run at speed T during $[0, 1]$, hence the total energy usage is T^α. Yet Opt can run at speed $\frac{T}{x}$ during $[0, x]$ to finish j_1, with total energy usage $(\frac{T}{x})^\alpha x$. Hence the competitive ratio is $\frac{T^\alpha}{(T/x)^\alpha(x)} = x^{\alpha-1}$
- If Alg has not completed j_1 at time 1, then another job j_2 is released at time 1 with $d(j_2) = x$, $p(j_2) = T(x-1)$, and $v(j_2) = T^\alpha\Gamma^{\alpha-1}(x-1)$. Note that $\tilde{s}(j_2) = T\Gamma$ and $\Gamma(j_2) = \Gamma$. Opt can complete both j_1 and j_2 by running at speed T throughout $[0, x]$, with total energy usage $T^\alpha x$. On the other hand, Alg cannot complete both j_1 and j_2 by their deadlines. If Alg discards j_1, the competitive ratio is at least $\frac{v(j_1)}{T^\alpha x} = \frac{\Gamma^{\alpha-1}}{x}$; if Alg discards j_2, the competitive ratio is at least $\frac{v(j_2)}{T^\alpha x} = \Gamma^{\alpha-1} - \frac{\Gamma^{\alpha-1}}{x}$.

Note that $\Gamma(j_1) = \Gamma(j_2) = \Gamma$, so the penalty ratio of the input sequence is Γ. The competitive ratio is at least $k = \min\{x^{\alpha-1}, \frac{\Gamma^{\alpha-1}}{x}, \Gamma^{\alpha-1} - \frac{\Gamma^{\alpha-1}}{x}\}$. If $\Gamma \geq 2^{\frac{\alpha}{\alpha-1}}$, we set $x = \Gamma^{\frac{\alpha-1}{\alpha}}$, then $x \geq 2$, $\Gamma^{\alpha-1} = x^\alpha \geq 2x^{\alpha-1}$ and $k \geq x^{\alpha-1} = \Gamma^{\alpha-2+1/\alpha}$. If $\Gamma < 2^{\frac{\alpha}{\alpha-1}}$, we set $x = 2$ and $k \geq \frac{1}{2}\Gamma^{\alpha-1}$. □

Note that $\frac{1}{2}\Gamma^{\alpha-1} = \Omega(\Gamma^{\alpha-2+1/\alpha})$. When T is large, the $e^{\alpha-1}/\alpha$ lower bound from the unbounded speed model carries to the bounded speed model. Hence, we have the following.

Corollary 9. *For bounded speed model, any algorithm is* $\Omega(\max\{e^{\alpha-1}/\alpha, \Gamma^{\alpha-2+1/\alpha}\})$ *-competitive.*

4.2 The Algorithm BPS

We propose an algorithm BPS(c), which stands for Bounded Profitable Speed with parameter c and is a natural extension to the PS algorithm. We will show that it is $O(\alpha^\alpha + 2\Gamma^{\alpha-1}(\alpha+1)^{\alpha-1})$-competitive. BPS($c$) maintains a list Q of admitted jobs, which is empty initially and is maintained as follows.

Algorithm BPS(c)

- **Job execution.** At any time t, BPS(c) uses OA to schedule the jobs in Q.
- **Job admission.** When a job j arrives at $r(j)$, let S be the set of jobs remaining in Q just before j arrives. BPS(c) calculates the OA schedule for $S \cup \{j\}$. Let $s(j)$ be the speed of j in this OA schedule. BPS(c) admits j into Q if $s(j) \leq \min\{c \cdot \tilde{s}(j), T\}$; and j is discarded immediately otherwise.
- **Job completion.** When a job j in Q is completed, remove it from Q.

In our analysis, $c = 1$ gives the best competitive ratio for BPS(c). To ease our discussion, we will fix $c = 1$ and call the resulting algorithm BPS. We first observe that BPS is a valid

algorithm, i.e., it uses a speed at most T at any time, as follows. Note that the speed of BPS may increase only when a job j is admitted. Let S be the set of jobs remaining in Q just before j arrives. We can obtain the OA schedule for $S \cup \{j\}$ from that for S by imagining the size of j to be increasing from zero to $p(j)$. Note that the density of the critical interval containing $d(j)$ will be increasing, and it may split into multiple ones, or merge with other intervals. Since j is admitted eventually, its final density is at most $\min\{\tilde{s}(j), T\} \leq T$. All other critical intervals with larger densities are not affected. Hence, the maximum speed of the OA schedule for $S \cup \{j\}$ remains at most T.

Note that if $\Gamma \leq 1$, then $\min\{\tilde{s}(j), T\} = \min\{\tilde{s}(j), \tilde{s}(j)/\Gamma(j)\} = \tilde{s}(j)$. Hence BPS and PS(1) will admit the same set of jobs and consequently have the identical schedule. Note that by putting $c = 1$ and $b = \alpha + 1$ into Theorem 1, PS(1) is $O(\alpha^\alpha)$-competitive in the unbounded speed model, which implies that BPS is $O(\alpha^\alpha)$-competitive in the bounded speed model. In the following, we assume $\Gamma > 1$. Our main result in this subsection is the following theorem.

Theorem 10. *Let $\Gamma > 1$ be the penalty ratio. BPS is $\left((1 + \Gamma^{\alpha-1}\frac{b^{\alpha-1}}{(b-1)^\alpha}) \max\{\alpha^\alpha, \alpha^2\} + \Gamma^{\alpha-1} b^{\alpha-1}\right)$-competitive in the bounded speed model, for any $b > 1$.*

Putting $b = \alpha + 1$ for $\alpha \geq 2$, and $b = \alpha^{2/\alpha} + 1$ for $1 < \alpha < 2$, we have the following corollary.

Corollary 11. *Let $\Gamma > 1$ be the penalty ratio. For $1 < \alpha < 2$, BPS is $\left(\alpha^2 + 2\Gamma^{\alpha-1}(\alpha^{2/\alpha} + 1)^{\alpha-1}\right)$-competitive; for $\alpha \geq 2$, BPS is $\left(\alpha^\alpha + 2\Gamma^{\alpha-1}(\alpha + 1)^{\alpha-1}\right)$ -competitive.*

The rest of this subsection proves Theorem 10. Consider any job sequence J' with penalty ratio $\Gamma > 1$. Let Opt$'$ be the optimal offline algorithm in the bounded speed model. Let E'_a and E'_o be the energy usage of BPS and Opt$'$, respectively. Let D'_a and D'_o be the value discarded by BPS and Opt$'$, respectively. We first analyze D'_a. The proof is similar to that of Lemma 3.

Lemma 12. *$D'_a \leq \Gamma^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha} E'_a + b^{\alpha-1} E'_o\right) + D'_o$, for any $b > 1$.*

Proof. Let J'_{D1} be the set of jobs discarded by BPS yet completed by Opt$'$. It suffices to show that $v(J'_{D1}) \leq \Gamma^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha} E'_a + b^{\alpha-1} E'_o\right)$. Let j be an arbitrary job in J'_{D1}. Let $I'(j)$ be the set of maximal time intervals during which Opt$'$ processes j. Denote $|I'(j)|$ as the total length of the intervals in $I'(j)$. Denote $E'_a(I'(j))$ and $E'_o(I'(j))$ as the total energy usage by BPS and Opt$'$ during $I'(j)$, respectively. We first bound $v(j)$ by $E'_a(I'(j))$ and $E'_o(I'(j))$.

If $\Gamma(j) < 1$, we can check that by repeating the argument of Lemma 3, we have $v(j) \leq \frac{b^{\alpha-1}}{(b-1)^\alpha} E'_a(I'(j)) + b^{\alpha-1} E'_o(I'(j)) \leq \Gamma^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha} E_a(I'(j)) + b^{\alpha-1} E_o(I'(j))\right)$ for any $b > 1$.

Otherwise, $\Gamma(j) \geq 1$, so $\tilde{s}(j) \geq T$. We define $\tilde{\ell}(j) = p(j)/\tilde{s}(j)$, $b'_j = |I'(j)|/\tilde{\ell}(j)$ and let $b''_j = b'_j/\Gamma(j)$. Since Opt$'$ completes $p(j)$ units of work in $I'(j)$, similar to Lemma 3, $E'_o(I'(j)) = (\frac{p(j)}{|I'(j)|})^\alpha |I'(j)| = \frac{v(j)}{(b'_j)^{\alpha-1}} = \frac{1}{(\Gamma(j))^{\alpha-1}} \frac{1}{(b''_j)^{\alpha-1}} v(j)$. Since BPS discards j, we can show that work done by BPS during $I'(j)$ is at least $T|I'(j)| - p(j)$. Then, $E'_a(I'(j)) \geq (\frac{T|I'(j)| - p(j)}{|I'(j)|})^\alpha |I'(j)| = \frac{((b'_j)/\Gamma(j) - 1)^\alpha}{(b'_j)^{\alpha-1}} v(j) = \frac{1}{(\Gamma(j))^{\alpha-1}} \frac{(b''_j - 1)^\alpha}{(b''_j)^{\alpha-1}} v(j)$. Note that $b'_j = \frac{|I'(j)|}{\tilde{\ell}(j)} \geq \frac{p(j)}{T\tilde{\ell}(j)} = \frac{\tilde{s}(j)}{T} = \Gamma(j)$ and hence $b''_j \geq 1$. Furthermore, $\frac{(b''_j - 1)^\alpha}{(b''_j)^{\alpha-1}}$ is increasing for $b''_j \geq 1$. Hence, similar to Lemma 3, for any $b > 1$ we can consider the cases of $b > b''_j$ and $b \leq b''_j$ to obtain

$$v(j) \leq (\Gamma(j))^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha} E_a(I'(j)) + b^{\alpha-1} E_o(I'(j))\right) \leq \Gamma^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha} E_a(I'(j)) + b^{\alpha-1} E_o(I'(j))\right)$$

Summing up the bound on $v(j)$ over all $j \in J'_{D1}$ and note that $I'(j)$ are disjointed, the lemma follows. □

We show an upper bound for E'_a as follows.

Lemma 13. $E'_a \leq \max\{\alpha^\alpha, \alpha^2\}(E'_o + D'_o)$.

Proof. To ease the discussion, denote $E(\text{Alg}, J')$ as the total energy usage of algorithm Alg on an input sequence J'. and $cost(\text{Alg}, J')$ as the total energy usage plus value discarded of Alg on input J'. Recall that Opt and Opt$'$ are the optimal offline algorithms in the unbounded and bounded speed models, respectively. Note that $E'_a = E(\text{BPS}, J')$ and $E'_o + D'_o = cost(\text{Opt}', J')$.

For any input sequence J', let J'_a be the set of jobs completed by BPS. Note that the schedule of BPS on input J' is identical to the schedule of BPS on input J'_a. Therefore, we have $E(\text{BPS}, J') = E(\text{BPS}, J'_a) = E(\text{PS}(1), J'_a) \leq \max\{\alpha^\alpha, \alpha^2\} cost(\text{Opt}, J'_a)$, where the second equality comes from the definitions of BPS and PS(1), and the inequality follows from Lemma 4. Note that $cost(\text{Opt}, J'_a) \leq cost(\text{Opt}', J'_a) \leq cost(\text{Opt}', J')$, where the first inequality holds because the optimal algorithm in the unbounded speed model is no worse than that in the bounded speed model, and the second inequality holds because the optimal cost will not increase with more jobs. The lemma follows. □

Finally, Theorem 10 follows directly from Lemmas 12 and 13.

References

[1] S. Albers. Energy-efficient algorithms. *Communications ACM*, 53(5):86–96, 2010.

[2] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, 3(4), 2007.

[3] L. Andrew, A. Wierman, and A. Tang. Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):39–41, 2009.

[4] N. Bansal, H. L. Chan, T. W. Lam, and L. K. Lee. Scheduling for bounded speed processors. In *Proc. of International Colloquium on Automata, Languages and Programming, ICALP*, pages 409 – 420, 2008.

[5] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA*, pages 693–701, 2009.

[6] N. Bansal, H.-L. Chan, K. Pruhs, and D. Rogozhnikov-Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. In *ICALP(1)*, pages 144–155, 2009.

[7] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *JACM*, 54(1), 2007.

[8] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–813, 2007.

[9] C. Belady. In the data center, power and cooling costs more than the it equipment it supports. *Electronics Cooling Magazine*, 13(1):24–27, 2007. http://electronics-cooling.com/articles/2007/feb/a3/.

[10] 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 *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 795–804, 2007.

[11] G. Koren and D. Shasha. D^{over}: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM J. Comput.*, 24(2):318–339, 1995.

[12] T. W. Lam, L. K. Lee, I. K. K. To, , and P. W. H. Wong. Speed scaling functions for flow time scheduling based on active job count. In *ESA*, pages 647–659, 2008.

[13] J. Markoff and S. Lohr. Intel's huge bet turns iffy. New York Times, September 29 2002.

[14] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Foundations of Computer Science (FOCS)*, pages 374–382, 1995.