Tradeoff between Energy and Throughput for Online Deadline Scheduling

Ho-Leung Chan[*], Tak-Wah Lam, and Rongbin Li

Department of Computer Science, University of Hong Kong
{hlchan,twlam,rbli}@cs.hku.hk

Abstract. We consider dynamic speed scaling on a single processor and study 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 scheduling algorithms to discard some of the jobs (i.e., not finishing them) 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 exists 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 a 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 [15]. 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 $P(s)$,

[*] Ho-Leung Chan is partially supported by HKU Seed Funding 201002159001.

K. Jansen and R. Solis-Oba (Eds.): WAOA 2010, LNCS 6534, pp. 59–70, 2011.

is modeled as s^{α}, where $\alpha > 1$ is a constant commonly believed to be 2 or 3 (determined by the physical properties of the hardware technology). Jobs with different sizes and deadlines arrive online over time. 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. [15] proposed two online algorithms AVR and OA, and showed that AVR is $(2^{\alpha-1}\alpha^{\alpha})$-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 which is $O(e^{\alpha})$-competitive (i.e., better than OA when 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 maximizes the throughput (which is the total size of jobs completed by their deadlines), and minimizes 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 on 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. Note that all the above studies assume throughput is the primary concern. That is, the objectives require a scheduling algorithm to first maximize the throughput and then minimize 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 a 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 any 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.

We first study the tradeoff in the unbounded speed model. Notice that the problem of minimizing the total energy usage plus 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* for each job, 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 usage and value discarded of the optimal schedule. Then we bound the energy usage of PS using a potential function 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 for 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 this singularity issue of getting a zero or close to zero value in the objective function. Recently and independently, Pruhs and Stein [14] studied the maximization objective. They consider the resource augmentation model where the online algorithm is given a processor that can run faster than that of the optimal with the same rate of energy usage; and they show that an $O(1)$-competitive algorithm exists.

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 its arrival 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 [8, 12, 5], Andrew et al. [3] have finally given a 2-competitive algorithm for minimizing energy plus flow time.

2 Preliminaries

We first define the problem formally and review the algorithm OA [15, 7].

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 $v(j)$. Let J and $v(J)$ denote a sequence of jobs and their total values. Preemption is allowed. 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, where s is the running speed and $\alpha > 1$ is a constant. Let $s(t)$ be the speed of the processor at time t. Then the total energy usage 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 $\sum_j s(j, t) \le s(t)$ for all t. A job j is completed by $d(j)$ if $\int_{r(j)}^{d(j)} s(j, t) dt \ge p(j)$; and j is 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.

Review of algorithm OA. 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 by EDF (earliest deadline first) and the speed during each critical interval $(t_{i-1}, t_i]$ is ρ_i. Note that ρ_i is decreasing. It can been shown that OA uses the minimum energy to complete S if no new jobs arrive. If a new job j arrives after time t, the OA schedule will be recomputed starting from the time $r(j)$. Below are some known properties about OA which will be used by our algorithms.

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 $c = \alpha^{(\alpha-2)/(\alpha-1)}$. First, 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. *The* profitable speed *of Job 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 is $(\tilde{s}(j))^\alpha \frac{p(j)}{\tilde{s}(j)} = (\tilde{s}(j))^{\alpha-1} p(j) = u(j)p(j) = v(j)$. □

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 $c \cdot \tilde{s}(j)$.

3.1 Algorithm PS(c)

Algorithm PS(c) (c is a parameter) 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 below.

Algorithm PS(c)

- **Job execution.** At any time, 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 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. *$\forall c > 0$, 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 conclude that PS($\alpha^{\frac{\alpha-2}{\alpha-1}}$) is $(\alpha^{\alpha} + 2e\alpha)$ -competitive.

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

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

Lemmas 1 and 2 together imply Theorem 1 as follows. For Opt, the total energy usage plus value discarded is $E_o + D_o$. Therefore, the total energy usage plus discard of PS(c) is

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

and Theorem 1 follows.

3.2 Value Discarded by PS(c)

This section analyzes D_a and proves Lemma 1. 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. $D_a = v(J_{D1}) + v(J_{D2}) \leq v(J_{D1}) + D_o$. To prove Lemma 1, it is sufficient to show that $v(J_{D1}) \leq \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a + b^{\alpha-1} E_o$.

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 3. *Let j be any job in J_{D1}, then $v(j) \leq \frac{b^{\alpha-1}}{(cb-1)^\alpha} E_a(I(j)) + b^{\alpha-1} E_o(I(j))$ 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,

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

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,

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

Finally, for $b > 1/c$, there are two cases. If $b > b_j$, 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 $f(x) = \frac{x^{\alpha-1}}{(cx-1)^\alpha}$ decreases when $x > \frac{1}{c}$. Hence for all $b > \frac{1}{c}$, Lemma 3 holds. □

Next, 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 3 over all jobs in J_{D1}, we obtain $v(J_{D1}) \leq \frac{b^{\alpha-1}}{(cb-1)^\alpha}E_a + b^{\alpha-1}E_o$. Hence, Lemma 1 follows immediately.

3.3 Energy Usage of PS(c)

This section analyzes E_a and proves Lemma 2. 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 arrival and after all deadlines.
- *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], we can then prove by induction on time that

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

which implies Lemma 2 as $\Phi(t) = 0$ after all deadlines are passed.

Definition of the potential function. Consider any time t. For any $t'' \geq t' \geq t$, 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

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

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, $\Delta\Phi(t) \leq 0$. On the other hand, $\Delta D_o(t) \geq 0$ 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 4. *Let $\Delta\Phi(t)$ be the change in $\Phi(t)$ if j is admitted by PS(c) and discarded by Opt. Then $\Delta\Phi(t) \leq \alpha^2 c^{\alpha-1} v(j)$.*

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

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

Note that for any convex function $f(z)$ and any real numbers $y > x$, we have $f(y) - f(x) \leq f'(y)(y - x)$, 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

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

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

Therefore, if Opt discards j, $\Delta D_o = v(j)$, so $\Delta\Phi(t) \leq \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}\Delta D_o$. 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 5. Hence, $\Delta\Phi(t) \leq 0 = \max\{\alpha^\alpha, \alpha^2 c^{\alpha-1}\}\Delta D_o$.

Lemma 5. *([7]) Let $q, r, \delta \geq 0$ and $\alpha \geq 1$, then $(q+\delta)^{\alpha-1}(q+\delta-\alpha(r+\delta)) - q^{\alpha-1}(q-\alpha r) \leq 0$.*

So far, we assume that $p(j)$ is small and only one critical interval is affected. If $p(j)$ is large, we follow the technique of [7, 10]. We 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. $p(j_2) = p(j) - p(j_1)$. 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 [7]. 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 $(s_o(t))^\alpha$ while $D_o(t)$ remains constant. Note that to prove the running condition, it is sufficient to prove that $(s_a(t))^\alpha + \frac{d}{dt}\Phi(t) - \alpha^\alpha(s_o(t))^\alpha \leq 0$. 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 processes jobs by EDF. So at time t, PS(c) is processing a job with deadline in I_1. $s_a = \rho_1$, so $w_a(t_0, t_1)$ is decreasing at a rate of s_a. Suppose Opt is processing a job with deadline in I_k, where $k \geq 1$. Then $w_o(t_{k-1}, t_k)$ is decreasing at a rate of s_o. Therefore $\frac{d}{dt}\Phi = \alpha\rho_1^{\alpha-1}(-s_a) + \alpha^2\rho_k^{\alpha-1}s_o \leq \alpha\rho_1^{\alpha-1}(-s_a) + \alpha^2\rho_1^{\alpha-1}s_o = -\alpha s_a^\alpha + \alpha^2 s_a^{\alpha-1}s_o$, where the inequality comes from $\rho_k \leq \rho_1$. Finally, $s_a^\alpha + \frac{d}{dt}\Phi - \alpha^\alpha s_o^\alpha \leq (1-\alpha)s_a^\alpha + \alpha^2 s_a^{\alpha-1}s_o - \alpha^\alpha s_o^\alpha$. The last expression can be shown to be non-positive by differentiation.

4 Bounded Speed Model

We first define the penalty ratio of a job sequence.

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

Theorem 2. *For the bounded speed model, any algorithm has competitive ratio at least $\min\{\Gamma^{\alpha-2+1/\alpha}, \frac{1}{2}\Gamma^{\alpha-1}\}$, where Γ is the penalty ratio of the 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, Alg must run at speed T during $[0, 1]$. The total energy usage is T^α. Opt can run at speed $\frac{T}{x}$ during $[0, x]$ to finish j_1, with total energy usage $(\frac{T}{x})^\alpha x$. So the competitive ratio is $\frac{T^\alpha}{(T/x)^\alpha (x)} = x^{\alpha-1}$
- If Alg has not completed j_1 at time 1, 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 $T^\alpha x$. 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, it 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 holds. Hence, for bounded speed model, any algorithm is $\Omega(\max\{e^{\alpha-1}/\alpha, \Gamma^{\alpha-2+1/\alpha}\})$ -competitive.

4.2 Algorithm BPS

We propose an algorithm BPS(c) (Bounded Profitable Speed with parameter c). We 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 maintained as follows.

Algorithm BPS(c)

- **Job execution.** At any time, 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 is completed, remove it from Q.

In our analysis, $c = 1$ gives the best competitive ratio for BPS(c), hence, to ease our discussion, we will fix $c = 1$ and call the resulting algorithm BPS. Note that if $\Gamma \leq 1$, then $\min\{\tilde{s}(j), T\} = \tilde{s}(j)$, and then BPS and PS(1) will admit the same set of jobs and consequently have the identical schedule. 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 for $\Gamma \leq 1$. Hence, in the following, we assume $\Gamma > 1$.

Theorem 3. *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* $b > 1$*, where* $\Gamma > 1$ *is the penalty ratio.*

Putting $b = \alpha + 1$ for $\alpha \geq 2$, and $b = \alpha^{2/\alpha} + 1$ for $1 < \alpha < 2$, we conclude that BPS is $\left(\alpha^2 + 2\Gamma^{\alpha-1}(\alpha^{2/\alpha} + 1)^{\alpha-1}\right)$-competitive for $1 < \alpha < 2$, and $\left(\alpha^\alpha + 2\Gamma^{\alpha-1}(\alpha + 1)^{\alpha-1}\right)$ -competitive for $\alpha \geq 2$. To prove Theorem 3, consider any job sequence J'. 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. Theorem 3 follows from the following two inequalities. The proofs are left to the full paper.

- $D'_a \leq \Gamma^{\alpha-1}\left(\frac{b^{\alpha-1}}{(b-1)^\alpha}E'_a + b^{\alpha-1}E'_o\right) + D'_o$, for any $b > 1$.
- $E'_a \leq \max\{\alpha^\alpha, \alpha^2\}(E'_o + D'_o)$.

References

1. Albers, S.: Energy-efficient algorithms. ACM Communications 53(5), 86–96 (2010)
2. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4) (2007)
3. Andrew, L., Wierman, A., Tang, A.: Optimal speed scaling under arbitrary power functions. ACM SIGMETRICS Performance Evaluation Review 37(2), 39–41 (2009)
4. Bansal, N., Chan, H.L., Lam, T.W., Lee, L.K.: Scheduling for bounded speed processors. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 409–420. Springer, Heidelberg (2008)
5. Bansal, N., Chan, H.-L., Pruhs, K.: Speed scaling with an arbitrary power function. In: SODA, pp. 693–701 (2009)
6. Bansal, N., Chan, H.-L., Pruhs, K., Katz, D.: Improved bounds for speed scaling in devices obeying the cube-root rule. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 144–155. Springer, Heidelberg (2009)
7. Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. JACM 54(1) (2007)
8. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 805–813 (2007)
9. Belady, C.: 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. Chan, H.L., Chan, W.T., Lam, T.W., Lee, L.K., Mak, K.S., Wong, P.W.H.: Energy efficient online deadline scheduling. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 795–804 (2007)
11. Koren, G., Shasha, D.: D$^{\text{over}}$: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. SIAM J. Comput. 24(2), 318–339 (1995)
12. Lam, T.W., Lee, L.K., To, I.K.K., Wong, P.W.H.: Speed scaling functions for flow time scheduling based on active job count. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 647–659. Springer, Heidelberg (2008)
13. Markoff, J., Lohr, S.: Intel's huge bet turns iffy. New York Times (September 29, 2002)
14. Pruhs, K., Stein, C.: How to schedule when you have to buy your energy. To appear in RANDOM-APPROX (2010)
15. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Foundations of Computer Science (FOCS), pp. 374–382 (1995)