Online Deadline Scheduling with Bounded Energy Efficiency

Joseph Wun-Tat Chan[1], Tak-Wah Lam[⋆,2], Kin-Sum Mak[2], and Prudence W.H. Wong[⋆⋆,3]

[1] Department of Computer Science, King's College London, UK
joseph.chan@kcl.ac.uk
[2] Department of Computer Science, The University of Hong Kong, Hong Kong
{twlam, ksmak}@cs.hku.hk
[3] Department of Computer Science, University of Liverpool, UK
pwong@csc.liv.ac.uk

Abstract. Existing work on scheduling with energy concern has focused on minimizing the energy for completing all jobs or achieving maximum throughput [19, 2, 7, 13, 14]. That is, energy usage is a secondary concern when compared to throughput and the schedules targeted may be very poor in energy efficiency. In this paper, we attempt to put energy efficiency as the primary concern and study how to maximize throughput subject to a user-defined threshold of energy efficiency. We first show that all deterministic online algorithms have a competitive ratio at least Δ, where Δ is the max-min ratio of job size. Nevertheless, allowing the online algorithm to have a slightly poorer energy efficiency leads to constant (i.e., independent of Δ) competitive online algorithm. On the other hand, using randomization, we can reduce the competitive ratio to $O(\log \Delta)$ without relaxing the efficiency threshold. Finally we consider a special case where no jobs are "demanding" and give a deterministic online algorithm with constant competitive ratio for this case.

1 Introduction

Processor scheduling is a classical optimization problem. As the proliferation of mobile computing devices, energy usage has become an important performance measure for processor scheduling. *Dynamic voltage scaling* [9,15,18] is a technology that enables the reduction in energy usage. It allows a processor to run in variable speed; the rate of energy consumption of a processor running at speed s is believed to be s^α where $\alpha \geq 2$ [5]. Note that a processor running at speed s can do s units of work in one unit of time. To process a job with w units of work at speed s, a processor consumes $s^\alpha w/s = s^{\alpha-1}w$ units of energy. In other words, it is more energy efficient to schedule a job at a low speed whenever possible.

In the literature, the study of energy-efficient scheduling was mainly in the context of deadline scheduling [19,2,7,13,14]. Given a processor where jobs arrive

⋆ The research is supported partly by RGC Grant HKU-7140/06E.
⋆⋆ The research is supported partly by EPSRC Grant EP/E028276/1.

at unpredictable times with arbitrary work and deadline requirement, the aim is to design an (online) schedule that maximizes the throughput, which is the total work of the jobs completed by their deadlines. An algorithm A is said to be c-competitive, where $c \geq 1$, if for any job sequence, A produces a schedule with a throughput at least $1/c$ of the best offline schedule.

Existing work on scheduling with energy concern has focused on minimizing the energy for completing all jobs or achieving maximum throughput [19,2,7,13,14]. That is, energy usage is a secondary concern when compared to throughput and the schedules targeted may be very poor in energy efficiency. In this paper, we attempt to put energy efficiency as the primary concern and study how to maximize throughput subject to a user-defined threshold of energy efficiency. We define the *energy efficiency* of a schedule to be the total amount of work completed in time divided by the total energy usage. The range of efficiency is $[0, \infty)$. For example, assuming the processor completes every job (on time) it works on, using unit-speed gives an efficiency of 1; 0.5-speed gives an efficiency of 4 (assuming $\alpha = 3$); 2-speed gives an efficiency of 0.25; running the processor at high speed would give an efficiency approaching zero. We refer to an efficiency of 1 the “ideal” efficiency. We assume that the user has a preference on energy efficiency and would not accept schedules with energy efficiency below a certain threshold E. Given a job sequence and an *efficiency threshold* E, we aim at finding an energy-efficient schedule (i.e., with energy efficiency at least E) that maximizes the throughput. In this paper we investigate online algorithms that produce energy-efficient schedules with throughput competitive to the optimal schedule which has the maximum throughput while maintaining an energy efficiency at least E.

Previous work. There has been a number of work when throughput is the primary concern and energy usage the secondary. Yao et al. [19] considered the case where a processor can run at any speed $s \geq 0$. They gave two online algorithms AVR and OA, and showed that AVR is $2^\alpha \alpha^\alpha$-competitive on energy usage (against the optimal offline schedule that uses the minimum amount of energy to complete all jobs). OA was later proved, by Bansal et al. [2], to be α^α-competitive. Bansal et al. [2] further improved the result with a new algorithm that is $2(\alpha/(\alpha-1))^\alpha e^\alpha$-competitive. Chan et al. [7] considered the case where the speed of a processor is upper bounded by a constant T, i.e., the processor can run at any speed $s \in [0, T]$. The optimal schedule is the one that maximizes the throughput and minimizes the energy usage among all schedules with the maximum throughput. They proposed an online algorithm FSA(OAT) that is 14-competitive on throughput and $(\alpha^\alpha + \alpha^2 4^\alpha)$-competitive on energy with the optimal schedule. On the other hand, Li et al. [13] have considered structured jobs and shown that AVR has a better performance. Very recently, scheduling on a processor with a fixed number of discrete speed levels has also attracted some attention [12,14,7].

Our contributions. Assume that a processor can run at any speed $s \geq 0$, we study how to maximize throughput subject to a user defined threshold of energy efficiency. We first show that no deterministic online algorithm can achieve a constant competitive ratio on throughput while guarantee an energy efficiency at least E. Precisely, let Δ be the max-min ratio of work of the jobs. All deterministic online algorithms have a competitive ratio at least Δ. Then, we study the problem in three different directions.

1. **Relaxed energy efficiency threshold:** Assume that the energy efficiency threshold of the online algorithm is relaxed to $E/(1+\epsilon)$ for some constant $\epsilon > 0$, while that for the optimal offline algorithm remains E. We give a deterministic online algorithm with competitive ratio $2 + \frac{3}{(1+\epsilon)^{\frac{1}{\alpha-1}}-1}$. For example if $\alpha = 2$, the algorithm is $2 + 3/\epsilon$-competitive. Note that we have a lower bound that no deterministic online algorithm can achieve a competitive ratio of 1 even with relaxed energy efficiency threshold. However with limited pages, the details are omitted.
2. **Randomized algorithm:** We devise a randomized online algorithm with a competitive ratio of $O(\log \Delta)$. Our algorithm is adapted from the randomized algorithm of Goldman et al. [8]. They showed that their algorithm is $6(\log \Delta + 1)$-competitive for a fixed-speed processor. The adapted algorithm works for a variable speed processor, which produces schedules with energy efficiency at least E and is still $O(\log \Delta)$-competitive.
3. **No demanding jobs:** We consider a special case that no job is *demanding.* A job is demanding if it could not be completed (by its deadline) by a processor running at unit speed. For this special case and with $E < 1$, we give a deterministic online algorithm with competitive ratio $2 + \frac{3}{1-E^{\frac{1}{\alpha-1}}}$. We can see that a smaller E would give a smaller competitive ratio. For $E \geq 1$, the lower bound of Δ for the general input also applies to this case.

Remarks. The literature also contains results on other interesting aspects of scheduling with energy concern [11]. Irani et al. [10] extended the result on AVR [19] to a setting where the processor has a sleep state, and showed that the extension increases the competitive ratio on energy by only a constant factor. On the other hand, Pruhs et al. [16] have studied the offline problem of minimizing total flow time subject to a fixed amount of energy, while Albers and Fujiwara [1] and Bansal et al. [4] have studied the online problem of minimizing a cost consisting of the energy usage and the total flow time. Furthermore, the offline problem of minimizing the makespan subject to a fixed amount of energy has been studied in [6, 17]. Another practical concern is the maximum temperature of the processor as the temperature is related to energy usage. Several interesting results have been reported in [2, 3].

Organization of paper. The rest of the paper is organized as follows. In Section 2, we give the problem definition and a general lower bound. In Section 3, we show that allowing the online algorithm to have a slightly poorer energy efficiency leads to a constant competitive online algorithm. In Section 4, we present the randomized algorithm and analyze its performance. Finally, in Section 5, we study the special case with no demanding jobs.

2 Preliminaries

For any job J, we denote the release time, work and deadline of J as $r(J)$, $w(J)$, and $d(J)$, respectively. We assume that all jobs satisfy the property $w(J) \leq (d(J) - r(J))/E^{\frac{1}{\alpha-1}}$, otherwise no algorithm can complete this individual job with energy efficiency at least E. The span of J, denoted as $\rho(J)$, is the time interval $[r(J), d(J)]$. For any set of jobs L, let $w(L)$ denote the total work of all jobs in L. To ease our discussion, we assume that an algorithm will never process a job after missing its deadline, and whenever we say that a job is completed, it is always meant to be completed by the deadline. We assume preemption is allowed, and a preempted job can be resumed at the point of preemption. Given an efficiency threshold E, the optimal schedule is one whose energy efficiency is at least E and the throughput is the maximum among these schedules. An online algorithm is said to be c-competitive if for any job sequence, the schedule produced has an energy efficiency at least E and throughput at least $1/c$ of the optimal schedule.

Next we give a lower bound on the competitive ratio on throughput.

Theorem 1. *To maintain a user defined energy efficiency E, any deterministic online algorithm is at least Δ-competitive on throughput, where Δ is the max-min ratio of the work of jobs.*

Proof. Consider an example with two jobs. The first job J_1 is released at time 0 with $d(J_1) = 2$ and $w(J_1) = 2/E^{\frac{1}{\alpha-1}}$. Any online algorithm must schedule this job immediately at its arrival with speed $1/E^{\frac{1}{\alpha-1}}$ in order to complete the job in time and satisfy the efficiency constraint. Otherwise, the adversary would stop releasing jobs. Then, at time 1, J_2 is released, with $d(J_2) = 2\Delta + 1$ and $w(J_2) = 2\Delta/E^{\frac{1}{\alpha-1}}$. The online algorithm cannot further schedule J_2 due to the efficiency constraint. However, the optimal schedule only schedules J_2, and hence the competitive ratio is Δ. $\square$

3 Relaxed Energy Efficiency Threshold

In this section we study the case where the energy efficiency threshold of the online algorithm is relaxed to $E/(1+\epsilon)$ for some constant $\epsilon > 0$, while that for the optimal offline algorithm remains E. We first present an algorithm named Efficiency, and show some properties of the optimal offline algorithm. Then we analyze the performance of this online algorithm.

3.1 The Online Algorithm

Efficiency$_E$ takes a parameter E and schedules jobs with speed $1/E^{\frac{1}{\alpha-1}}$. The selection of jobs to run depends on a notion called $wait(\cdot)$ defined as follows. For any job J, we denote by $lst(J)$ the latest start time of J such that J can still be finished by Efficiency$_E$, i.e., $lst(J) = d(J) - w(J)E^{\frac{1}{\alpha-1}}$. At any time t, we define the allowed waiting time of the job J, $wait(J,t) = lst(J) - t$.

> EFFICIENCY$_E$ works as follows: Whenever the processor is idle, EFFICIENCY$_E$ schedules the job with the smallest $wait(J,t) \geq 0$ with speed $1/E^{\frac{1}{\alpha-1}}$.

Lemma 1. *The energy efficiency of the schedule given by* EFFICIENCY$_E$ *is* E.

Proof. EFFICIENCY$_E$ always runs at speed $1/E^{\frac{1}{\alpha-1}}$ and it completes a job whenever the job is scheduled. Let T be the total time EFFICIENCY$_E$ works on I. Then the resulting efficiency is $\frac{T/E^{\frac{1}{\alpha-1}}}{T/E^{\frac{\alpha}{\alpha-1}}} = E$. □

We analyze the relation between the throughput of EFFICIENCY$_E$ and the optimal offline algorithm OPT. Consider any job sequence I. Let $A \subseteq I$ be the set of jobs scheduled by EFFICIENCY$_E$ and let $N = I - A$. Consider the union of spans of all jobs in N, i.e., $\bigcup_{X \in N} \rho(X)$. Let $\ell = |\bigcup_{X \in N} \rho(X)|$. We show below that the competitive ratio of EFFICIENCY$_E$ depends on the ratio $\ell/w(A)$. Consider the optimal schedule for I. We denote by S, M, F the amount of work finished by OPT using slow, medium and fast speed, respectively. Precisely, S, M, F denote the amount of work OPT finishes with speed $\leq 1/E^{\frac{1}{\alpha-1}}$, $\leq 2/E^{\frac{1}{\alpha-1}}$, and $> 2/E^{\frac{1}{\alpha-1}}$, respectively. Note that the total work finished by OPT equals $M+F$. The following lemma gives the relation between S, M, F and $w(A)$.

Lemma 2. *(i)* $F \leq S$*; (ii)* $M \leq w(A) + 2\ell/E^{\frac{1}{\alpha-1}}$*; (iii)* $S \leq w(A) + \ell/E^{\frac{1}{\alpha-1}}$.

Proof. (i) Suppose the periods OPT running with speed $> 2/E^{\frac{1}{\alpha-1}}$ have durations $t_1, t_2, \ldots, t_n$ and speed $2k_1/E^{\frac{1}{\alpha-1}}, 2k_2/E^{\frac{1}{\alpha-1}}, \ldots, 2k_n/E^{\frac{1}{\alpha-1}}$, respectively, for some $k_i > 1$. Since the efficiency of the resulting schedule of OPT must be at least E, we have

$$\frac{S + \sum_{1 \leq i \leq n} 2k_i t_i / E^{\frac{1}{\alpha-1}}}{\sum_{1 \leq i \leq n} 2^\alpha k_i^\alpha t_i / E^{\frac{\alpha}{\alpha-1}}} \geq \text{ Energy efficiency of OPT } \geq E\,.$$

Thus, we have $\sum_{1 \leq i \leq n} (2^{\alpha-1} k_i^{\alpha-1} - 1)(2k_i t_i / E^{\frac{1}{\alpha-1}}) \leq S$. Since $(2^{\alpha-1} k_i^{\alpha-1} - 1) \geq 1$, we have $F = \sum_{1 \leq i \leq n} (2k_i t_i / E^{\frac{1}{\alpha-1}}) \leq S$.

(ii) Using speed at most $2/E^{\frac{1}{\alpha-1}}$, OPT can at most complete all jobs in A and a work of $2\ell/E^{\frac{1}{\alpha-1}}$ during the period of length ℓ. Therefore, $M \leq w(A) + 2\ell/E^{\frac{1}{\alpha-1}}$.

(iii) Similarly, using speed at most $1/E^{\frac{1}{\alpha-1}}$, OPT can at most complete all jobs in A and a work of $\ell/E^{\frac{1}{\alpha-1}}$ during the period of length ℓ. Therefore, $S \leq w(A) + \ell/E^{\frac{1}{\alpha-1}}$. □

Lemma 3. $\frac{OPT}{w(A)} \leq 2 + \frac{3\ell}{w(A)} \cdot \frac{1}{E^{\frac{1}{\alpha-1}}}$.

Proof. $OPT = M + F \leq M + S$, the inequality is due to Lemma 2 (i). Together with Lemma 2 (ii) and (iii), the lemma follows. □

3.2 Performance of $\textsc{Efficiency}_{E/(1+\epsilon)}$

We consider the algorithm $\textsc{Efficiency}_{E/(1+\epsilon)}$ and bound the ratio $\frac{\ell}{w(A)}$, thereby, show that $\textsc{Efficiency}_{E/(1+\epsilon)}$ is constant competitive.

For any $X \in N$, X is not scheduled by Efficiency, implying that Efficiency has scheduled some other jobs during the span $\rho(X)$. The following lemma gives a lower bound on $w(r(X), b)$, which denotes the amount of work finished by Efficiency during the interval $[r(X), b] \subseteq \rho(X)$.

Lemma 4. *For any $X \in N$, $w(r(X), t) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X))/E^{\frac{1}{\alpha-1}}$, for any $r(X) < t \le d(X)$.*

Proof. From $r(X)$ to $lst(X)$, Efficiency must not be idle. Since the processor runs at a speed $((1+\epsilon)/E)^{\frac{1}{\alpha-1}}$, at any time $r(X) < t \le lst(X)$, it has completed a work of $((1+\epsilon)/E)^{\frac{1}{\alpha-1}}(t - r(X)) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X))/E^{\frac{1}{\alpha-1}}$.

Furthermore, we have

$$\begin{aligned} w(r(X), lst(X)) &= ((1+\epsilon)/E)^{\frac{1}{\alpha-1}}(lst(X) - r(X)) \\ &= ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)|\rho(X)|/E^{\frac{1}{\alpha-1}} + |\rho(X)|/E^{\frac{1}{\alpha-1}} \\ &\quad -(d(X) - lst(X))((1+\epsilon)/E)^{\frac{1}{\alpha-1}} \\ &= ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)|\rho(X)|/E^{\frac{1}{\alpha-1}} + |\rho(X)|/E^{\frac{1}{\alpha-1}} - w(X). \end{aligned}$$

By the assumption of the work of a job, we have $w(X) \le |\rho(X)|/E^{\frac{1}{\alpha-1}}$. Hence, $w(r(X), lst(X)) \ge ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)|\rho(X)|/E^{\frac{1}{\alpha-1}} \ge ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X))/E^{\frac{1}{\alpha-1}}$ for $r(X) < t \le d(X)$. Since X is not scheduled by Efficiency, another job must be scheduled at $lst(X)$ and last until a time after $lst(X)$ and by Efficiency it must complete, and thus $w(r(X), t) > w(r(X), lst(X))$ for any $t > lst(X)$. Thus, we have $w(r(X), t) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X))/E^{\frac{1}{\alpha-1}}$ for $lst(X) < t \le d(X)$. This completes the proof of the lemma. □

Based on Lemma 4, we can bound $w(A)/\ell$ as follows.

Lemma 5. $w(A) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)\ell/E^{\frac{1}{\alpha-1}}$.

Proof. Let M be a minimal subset of N such that $\bigcup_{X \in M} \rho(X) = \bigcup_{X \in N} \rho(X)$, i.e., M induces the same union of spans as N does. Note that no job in M has its span being a sub-interval of any job in M and no three jobs in M have their spans overlapping at a common time. A consequence is that we can arrange the jobs in M such that the arrival times, as well as the deadlines, of the jobs are strictly increasing. Furthermore, the span of the jobs may form several disjoint continuous intervals $\rho_1, \rho_2, \cdots, \rho_k$, for some k. To prove the lemma, it suffices to show that $w(\rho_i) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)|\rho_i|/E^{\frac{1}{\alpha-1}}$ for all $1 \le i \le k$.

Suppose the subset of jobs in M with span in ρ_i is $\{X_1, X_2, \cdots, X_m\}$ such that $r(X_j) < r(X_{j+1}) < d(X_j) < d(X_{j+1})$ for all $1 \le j \le m-1$. We are going to show by induction on j that, for any $r(X_1) < t \le d(X_j)$, $w(r(X_1), t) >$

$((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X_1))/E^{\frac{1}{\alpha-1}}$. By Lemma 4, the base case is true. Assume it is true up to some j. Consider X_{j+1}. By induction and $r(X_{j+1}) \le d(X_j)$, we have $w(r(X_1), t) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X_1))/E^{\frac{1}{\alpha-1}}$ for $r(X_1) < t \le r(X_{j+1})$. Using Lemma 4 for X_{j+1} and $r(X_{j+1}) < t \le d(X_{j+1})$, we have, for $r(X_1) < t \le d(X_{j+1})$, $w(r(X_1), t) > ((1+\epsilon)^{\frac{1}{\alpha-1}} - 1)(t - r(X_1))/E^{\frac{1}{\alpha-1}}$. Therefore, the hypothesis is true for $j+1$ and the lemma follows. □

With Lemmas 3 and 5 , we have a constant competitive ratio for Efficiency$_{E/(1+\epsilon)}$.

Theorem 2. Efficiency$_{E/(1+\epsilon)}$ *is* $2 + \frac{3}{(1+\epsilon)^{\frac{1}{\alpha-1}} - 1}$*-competitive on throughput when the energy efficiency threshold is relaxed to* $E/(1+\epsilon)$*.*

4 Randomized Algorithm

As we have seen in Theorem 1, any deterministic online algorithm is at least Δ-competitive on throughput, where Δ is the max-min ratio of the work of jobs. In this section we show that randomization helps to overcome this barrier. By adapting the randomized algorithm of Goldman et al. [8], we can yield a competitive ratio of $O(\lceil \log \Delta \rceil)$ on throughput while maintaining an energy efficiency of E.

This section proceeds as follows. We first present the adapted randomized algorithm called Ran and state the results of Goldman et al. which also apply to Ran when the processor speed is fixed. Then, in Section 4.1, we analyze the competitive ratio of Ran when the processor speed can vary. More interestingly, we compare Ran with both the optimal non-preemptive and preemptive energy-efficient schedules (i.e., with energy efficiency at least E). A crucial lemma we used in the analysis would be proved in Section 4.2.

The algorithm of Goldman et al. schedules jobs in a processor with fixed unit speed. Each job is either (1) scheduled, (2) virtually scheduled, or (3) rejected. The job only runs if it is scheduled. A *virtually scheduled* job J does not run itself, but prevents any job of work less than $2w(J)$ from running. Thus, virtually scheduling a job J holds the processor for a longer job with a short wait time that may arrive during the interval when J is virtually scheduled. Assume the work of jobs is in $[1, \Delta]$. A number of queues are maintained each for jobs with different work. The queue for jobs of work in $(2^\ell, 2^{\ell+1}]$ is denoted by Q_ℓ. The adapted randomized algorithm Ran is given as follows. We assume that the processor runs at speed $1/E^{\frac{1}{\alpha-1}}$.

```
When a job $J$ arrives
    Suppose the work of $J$, i.e., $w(J)$, is in $(2^{\ell-1}, 2^\ell]$.
    If the system is idle, or if another job $J'$ is virtually scheduled
    where $w(J') \le 2^{\ell-1}$,
        With probability $\frac{1}{\lceil \log \Delta \rceil + 1 - \ell}$ we schedule $J$,
        otherwise virtually schedule $J$.
    Else place $J$ in $Q_\ell$.
```

```
When a scheduled or virtually scheduled job finishes at time $t$
    Let $Q_\ell$ be the non-empty queue with the largest $\ell$ possible.
        Let $J$ be the job having the smallest $wait(J,t)^1 \geq 0$ in $Q_\ell$.
        Remove $J$ from $Q_\ell$.
        With probability $\frac{1}{\lceil \log \Delta \rceil + 1 - \ell}$ we schedule $J$,
        otherwise virtually schedule $J$.
```

In the followings, we state a property of Ran and the performance of Ran against the optimal non-preemptive schedules at fixed speeds. Since Ran always schedules jobs at speed $1/E^{\frac{1}{\alpha-1}}$ and it completes each job it ever schedules, it always gives schedules with energy efficiency exactly E.

Fact 1. Ran *always gives schedules with energy efficiency exactly E.*

Let RAN_s be the expected throughput of Ran and OPT_s^n (resp. OPT_s^p) the throughput of the optimal non-preemptive (resp. preemptive) schedule for a processor with a fixed speed s. Goldman et al. [8] proved that OPT_s^n is at most $6(\lceil \log \Delta \rceil + 1)$ times of Ran$_s$.

Theorem 3 ([8]). $6(\lceil \log \Delta \rceil + 1)RAN_s \geq OPT_s^n$.

By slightly amending the analysis of Goldman et al., we can prove that the throughput of the optimal non-preemptive schedule with a fixed speed twice that of Ran, i.e., OPT_{2s}^n, is at most $8(\lceil \log \Delta \rceil + 1)$ times of RAN_s.

Theorem 4. $8(\lceil \log \Delta \rceil + 1)RAN_s \geq OPT_{2s}^n$.

4.1 Performance of the Randomized Algorithm

Although Ran always gives non-preemptive schedules, we analyze Ran in both non-preemptive and preemptive models, i.e., against the optimal non-preemptive and the optimal preemptive schedules with energy efficiency at least E.

Non-preemptive model. For a processor with variable speed, we show that the throughput of the optimal non-preemptive energy-efficient schedule is at most $14(\lceil \log \Delta \rceil + 1)$ times the expected throughput of the schedules produced by Ran. Recall from Section 3.1 that, in the optimal non-preemptive energy-efficient schedule, S and M denote the amount of work finished with speed $\leq 1/E^{\frac{1}{\alpha-1}}$ and $\leq 2/E^{\frac{1}{\alpha-1}}$, respectively. (Note that the results in Section 3.1 applies to both non-preemptive and preemptive models.) Moreover, by Lemma 2, the throughput of the optimal non-preemptive energy-efficient schedule is at most $S+M$. Obviously, S and M are no more than the maximum throughput using fixed speeds $1/E^{\frac{1}{\alpha-1}}$ and $2/E^{\frac{1}{\alpha-1}}$, respectively, with no energy concern. Thus, $S \leq OPT_s^n$ and $M \leq OPT_{2s}^n$ where $s = 1/E^{\frac{1}{\alpha-1}}$. Therefore, by Theorems 3 and 4, we have $S \leq 6(\lceil \log \Delta \rceil + 1)RAN_s$ and $M \leq 8(\lceil \log \Delta \rceil + 1)RAN_s$, and hence the throughput of the optimal non-preemptive energy-efficient schedule is at most $14(\lceil \log \Delta \rceil + 1)RAN_s$. We have the following theorem.

[1] Recall that $wait(J,t)$ is defined in Section 3.1 to be $lst(J) - t$ where $lst(J)$ is the latest start time of the job J such that J can still be completed on time.

Theorem 5. Ran *is* $14(\lceil \log \Delta \rceil + 1)$*-competitive in the non-preemptive model.*

Preemptive model. For a processor with variable speed, we show that the throughput of the optimal preemptive energy-efficient schedule is at most $70(\lceil \log \Delta \rceil + 1)$ times the expected throughput of the schedules produced by Ran. Similar to the non-preemptive case, in the optimal preemptive energy-efficient schedule, M and S denote the amount of work finished with speed $\leq 1/E^{\frac{1}{\alpha-1}}$ and $\leq 2/E^{\frac{1}{\alpha-1}}$, respectively. We can establish the relations $S \leq OPT_s^p$ and $M \leq OPT_{2s}^p$ where $s = 1/E^{\frac{1}{\alpha-1}}$ (recall that the superscript p refers to the optimal preemptive schedule).

The key to obtain the claimed competitive ratio of Ran in the preemptive model is to relate OPT_s^p and OPT_s^n, i.e., the throughput of the optimal preemptive and non-preemptive schedules of a processor at fixed speed s. We can show that $5OPT_s^n \geq OPT_s^p$ and we give the analysis in next section. We continue to derive the competitive ratio of Ran as follows. By Lemma 2, the throughput of the optimal preemptive energy-efficient schedule is at most $S + M$. Together with the relations $S \leq OPT_s^p$ and $M \leq OPT_{2s}^p$ where $s = 1/E^{\frac{1}{\alpha-1}}$, and $5OPT_{s'}^n \geq OPT_{s'}^p$ for any fixed speed s', we have the throughput of the optimal preemptive energy-efficient schedule at most $5OPT_s^n + 5OPT_{2s}^n$. Further applying Theorems 3 and 4, we can bound this optimal throughput by $70(\lceil \log \Delta \rceil + 1)RAN_s$. Hence, we have the following theorem.

Theorem 6. Ran *is* $70(\lceil \log \Delta \rceil + 1)$*-competitive in the preemptive model.*

4.2 Comparing Non-preemptive and Preemptive Optimal Schedules at Fixed Speed

We prove the claim $5OPT_s^n \geq OPT_s^p$ we used in the previous section, i.e., we show that, for any job sequence, with a processor at a fixed speed the throughput of the optimal preemptive schedule is at most 5 times that of the optimal non-preemptive schedule. In fact, we achieve this ratio by comparing the optimal preemptive schedule with a non-preemptive schedule $\mathcal{S}$ (not necessarily optimal) we constructed below.

Without loss of generality, we assume that the processor is at unit speed, i.e., $s = 1$. An (offline) non-preemptive schedule is constructed as follows. The idea is to mark a potential job that could be run in $\mathcal{S}$. At any time, at most one job is marked. A marked job may be unmarked, and further be marked again later. When a job has been marked continuously for a time period equal its work, we put the job in the non-preemptive schedule. The details of construction are given below.

```
When a job $J$ arrives
    If there is no marked job, or if another job $J'$ is marked
    with $w(J') \leq w(J)/2$ , unmark $J'$ and mark $J$.
When a job $J$ is marked continuously for $w(J)$ units of time
    Unmark $J$ and put $J$ in the non-preemptive schedule.
```

Mark the job J' with the largest $w(J')$ and $lst(J')$ not passed yet, if one exists.

The starting and finishing times of a job J in the schedule are the starting and ending times that the job is marked continuously for $w(J)$ units of time. It is clear that the schedule obtained above is non-preemptive since every job J in the schedule runs continuously for $w(J)$ units of time.

Before we compare the throughput of the non-preemptive schedule and the optimal preemptive schedule, we give some definitions and identify some properties of the non-preemptive schedule. Recall that $\mathcal{S}$ denotes the set of jobs in the non-preemptive schedule. For each job $J \in \mathcal{S}$, let $J_1, J_2, \ldots, J_k$ $(= J)$ be the sequence of jobs such that J_i is marked continuously for less than $w(J_i)$ time units and then unmarked because of the arrival and marking of J_{i+1} where $w(J_i) \leq w(J_{i+1})/2$, and finally J_k is marked continuously for $w(J_k)$ time units. Define an interval $I(J) = [a, b + 2w(J)]$ where a is the time that J_1 is initially marked in this sequence and b is the time that J_k $(= J)$ has been marked for $w(J_k)$ time units. We give some properties of the intervals as follows.

Fact 2. *For any time t if there is a job being marked at t, then $t \in I(J)$ for some job $J \in \mathcal{S}$.*

Lemma 6. *For any job $J \in \mathcal{S}$, the length of the interval $I(J)$ is at most $4w(J)$.*

Proof. Consider the sequence of jobs $J_1, J_2, \ldots, J_k$ $(= J)$ defined by J. We have $I(J) = [a, b + 2w(J)]$ where J_1 is initially marked at time a in this sequence and b is the time that J_k $(= J)$ has been marked for $w(J_k)$ $(= w(J))$ time units. By the definition, $b - a \leq w(J) + w(J)/2 + w(J)/2^2 + \ldots \leq 2w(J)$. Hence, we have $I(J) = b + 2w(J) - a \leq 4w(J)$. □

In the following lemma, we prove that the total work of all jobs in the optimal preemptive schedule but not in $\mathcal{S}$ is at most the sum of lengths of $I(J)$ for all jobs $J \in \mathcal{S}$. By this and the previous lemma, we can easily bound the throughput of the optimal preemptive schedule, as shown in Theorem 7.

Lemma 7. *The total work of all jobs in the optimal preemptive schedule but not in $\mathcal{S}$ is at most the sum of length of $I(J)$ for all jobs $J \in \mathcal{S}$.*

Proof. We prove the lemma by showing for all $J' \notin \mathcal{S}$ and for all $r(J') \leq t \leq d(J')$ that, $t \in I(J)$ for some job $J \in \mathcal{S}$. In other words, all the time that J' can be scheduled falls in the union of interval $I(J)$ for all $J \in \mathcal{S}$. Therefore, the total work of all jobs J' in the optimal preemptive schedule but not in $\mathcal{S}$ is bounded by the total length of the interval $I(J)$ for all $J \in \mathcal{S}$.

Let J' be a job in the optimal preemptive schedule but not in $\mathcal{S}$. First, we show for any time $r(J') \leq t \leq lst(J')$ that, $t \in I(J)$ for some $J \in \mathcal{S}$. Since $J' \notin \mathcal{S}$, there must be some job being marked during $[r(J'), lst(J')]$, as otherwise J' can be marked. Thus, by Fact 2, for any $r(J') \leq t \leq lst(J')$, we have $t \in I(J)$ for some job $J \in \mathcal{S}$.

Then, we also show for any time $lst(J') < t \leq d(J')$ that, $t \in I(J)$ for some $J \in \mathcal{S}$. Suppose $lst(J') \in I(X) = [a, b + 2w(X)]$ for the job $X \in \mathcal{S}$ where $I(X)$ has the maximum value of a. We claim that (i) $2w(X) > w(J')$ and (ii) $a \leq lst(J') \leq b$. Hence, for $lst(J') < t \leq d(J')$, we have $t \in [a, b + 2w(X)]$ as $d(J') - lst(J') = w(J') < 2w(X)$. Claim (i) is true because otherwise J' will be marked instead of X. Claim (ii) is true because if $b < lst(J') \leq b + 2w(X)$, J' or other jobs can be marked at $lst(J')$ and hence there exists another interval $I(Z) = [a', b' + 2w(Z)]$ that includes $lst(J')$ and $a' > a$, which is a contradiction. In conclusion, we have for any time $r(J') \leq t \leq d(J')$ that, $t \in I(J)$ for some $J \in \mathcal{S}$, and the lemma follows. □

Theorem 7. *The throughput of the optimal preemptive schedule is at most* 5 *times that of the optimal non-preemptive schedule, i.e.,* $5OPT_s^n \geq OPT_s^p$*, for a processor at a fixed speed* s *where* s *can be any constant greater than* 0*.*

Proof. By Lemma 7, the total work of jobs in the optimal preemptive schedule but not in $\mathcal{S}$ is at most the sum of length of $I(J)$ for all jobs $J \in \mathcal{S}$, which is at most 4 times the throughput of the non-preemptive schedule according to Lemma 6. Therefore, the throughput of the optimal preemptive schedule is at most 5 times the throughput of the non-preemptive schedule. As the throughput of the non-preemptive schedule must be at most that of the optimal non-preemptive schedule, the theorem follows. □

5 Without Demanding Jobs

We study the special case in which there are no demanding jobs that cannot be finished before deadline using unit speed. In other words, for every job J, $w(J) \leq d(J) - r(J)$. Because of the limited pages, we only the state the lemmas and theorems in this section without proofs. First we give a lower bound for this special case with no demanding jobs. The proof of which is similar to that of Theorem 1.

Theorem 8. *Without demanding jobs, if* $E \geq 1$*, any deterministic online algorithm is at least* Δ*-competitive on throughput, where* Δ *is the max-min ratio of the work of the jobs.*

Therefore, we only consider the case where $E < 1$ in the rest of this section. Without demanding jobs, we use the algorithm $\textsc{Efficiency}_E$ (as defined in Section 3), which by Lemma 1, gives schedules with energy efficiency at least E. By Lemma 3, we only need to bound the ratio $\ell/w(A)$ to obtain the competitive ratio of $\textsc{Efficiency}_E$. Recall the definition of I, A, N in Section 3. Similar to Lemma 4 and since jobs are not demanding, we can derive the following lemma.

Lemma 8. *For any* $X \in N$, $w(r(X), t) > (1/E^{\frac{1}{\alpha-1}} - 1)(t - r(X))$*, for any* $r(X) < t \leq d(X)$*.*

By repeating the argument for proving Lemma 5 and replacing the use of Lemma 4 by Lemma 8, we have the following lemma.

Lemma 9. $w(A) > (1/E^{\frac{1}{\alpha-1}} - 1)\ell$.

By Lemmas 3 and 9, we have a constant competitive ratio for $\textsc{Efficiency}_E$.

Theorem 9. *Without demanding jobs,* $\textsc{Efficiency}_E$ *is* $2+\frac{3}{1-E^{\frac{1}{\alpha-1}}}$*-competitive on throughput, for any efficiency threshold* $E < 1$.

References

1. S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. In *Proc. STACS*, pages 621–633, 2006.
2. N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *Proc. FOCS*, pages 520–529, 2004.
3. N. Bansal and K. Pruhs. Speed scaling to manage temperature. In *Proc. STACS*, pages 460–471, 2005.
4. N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. In *Proc. SODA*, pages 805–813, 2007.
5. D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.
6. D. P. Bunde. Power-aware scheduling for makespan and flow. In *Proc. SPAA*, pages 190–196, 2006.
7. 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 *Proc. SODA*, pages 795–804, 2007.
8. S. A. Goldman, J. Parwatikar, and S. Suri. Online scheduling with hard deadlines. *J. Algorithms*, 34(2):370–389, 2000.
9. D. Grunwald, P. Levis, K. I. Farkas, C. B. M. III, and M. Neufeld. Policies for dynamic clock scheduling. In *OSDI*, pages 73–86, 2000.
10. S. Irani, R. K. Gupta, and S. Shukla. Algorithms for power savings. In *Proc. SODA*, pages 37–46, 2003.
11. S. Irani and K. Pruhs. Algorithmic problems in power managment. *SIGACT News*, 32(2):63–76, 2005.
12. W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. *ACM Transactions on Embedded Computing Systems*, 4(1):211–230, Feb. 2005.
13. M. Li, B. J. Liu, and F. F. Yao. Min-energy voltage allocations for tree-structured tasks. In *Proc. COCOON*, pages 283–296, 2005.
14. M. Li and F. F. Yao. An efficient algorithm for computing optimal discrete voltage schedules. *SIAM J. Comput.*, 35(3):658–671, 2005.
15. P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP*, pages 89–102, 2001.
16. K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best resonse for your erg. In *Proc. SWAT04*, pages 15–25, 2004.
17. K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. In *Proc. WAOA*, pages 307–319, 2005.
18. M. Weiser, B. Welch, A. J. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *OSDI*, pages 13–23, 1994.
19. F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. FOCS*, pages 374–382, 1995.