THEORY OF COMPUTING, Volume 8 (2012), pp. 209–229
www.theoryofcomputing.org

SPECIAL ISSUE IN HONOR OF RAJEEV MOTWANI

Improved Bounds for Speed Scaling in Devices Obeying the Cube-Root Rule

Nikhil Bansal* Ho-Leung Chan Dmitriy Katz Kirk Pruhs†

Received: July 31, 2010; published: May 25, 2012.

Abstract: Speed scaling is a power management technology that involves dynamically changing the speed of a processor. This technology gives rise to dual-objective scheduling problems, where the operating system both wants to conserve energy and optimize some Quality of Service (QoS) measure of the resulting schedule. In the most investigated speed scaling problem in the literature, the QoS constraint is deadline feasibility, and the objective is to minimize the energy used. The standard assumption is that the processor power is of the form s^α where s is the processor speed, and $\alpha > 1$ is some constant; $\alpha \approx 3$ for CMOS based processors.

In this paper we introduce and analyze a natural class of speed scaling algorithms that we call qOA. The algorithm qOA sets the speed of the processor to be q times the speed that the optimal offline algorithm would run the jobs in the current state. When $\alpha = 3$, we show that qOA is 6.7-competitive, improving upon the previous best guarantee of 27 achieved by the algorithm Optimal Available (OA). We also give almost matching upper and lower bounds for qOA for general α. Finally, we give the first non-trivial lower bound, namely $\mathrm{e}^{\alpha-1}/\alpha$, on the competitive ratio of a general deterministic online algorithm for this problem.

ACM Classification: F.2.2

AMS Classification: 68Q25

Key words and phrases: scheduling, energy minimization, speed-scaling, online algorithms

*Part of this work was done while the author was at the IBM T. J. Watson Research Center.

†Supported in part by NSF grants CNS-0325353, CCF-0514058, IIS-0534531, and CCF-0830558, and an IBM Faculty Award.

 DOI: 10.4086/toc.2012.v008a009

1 Introduction

Current processors produced by Intel and AMD allow the speed of the processor to be changed dynamically. Intel's SpeedStep and AMD's PowerNOW technologies allow the operating system to dynamically change the speed of such a processor to conserve energy. In this setting, the operating system must not only have a *job selection policy* to determine which job to run, but also a *speed scaling policy* to determine the speed at which the job will be run. Almost all theoretical studies we know of assume a processor power function of the form $P(s) = s^{\alpha}$, where s is the speed and $\alpha > 1$ is some constant. Energy consumption is power integrated over time. The operating system is faced with a dual objective optimization problem as it both wants to conserve energy, and optimize some Quality of Service (QoS) measure of the resulting schedule.

The first theoretical study of speed scaling algorithms was in the seminal paper [16] by Yao, Demers, and Shenker. In the problem introduced in [16] the QoS objective was deadline feasibility, and the objective was to minimize the energy used. To date, this is the most investigated speed scaling problem in the literature [2, 6, 3, 9, 11, 12, 14, 16, 17]. In this problem, each job i has a release time r_i when it arrives in the system, a work requirement w_i, and a deadline d_i by which the job must be finished. The deadlines might come from the application, or might arise from the system imposing a worst-case quality-of-service metric, such as maximum response time or maximum slow-down. Since the speed can be made arbitrarily high, every job can always be completed by its deadline, and hence without loss of generality the job selection policy can be assumed to be Earliest Deadline First (EDF), as it produces a deadline feasible schedule whenever one exists. Thus the (only) issue here is to determine the processor speed at each time, i. e., find an online speed scaling policy, to minimize energy.

1.1 The story to date

In their seminal work, Yao, Demers, and Shenker [16] showed that the optimal offline schedule can be efficiently computed by a greedy algorithm YDS. They also proposed two natural online speed scaling algorithms, Average Rate (AVR) and Optimal Available (OA). Conceptually, AVR is oblivious in that it runs each job in the way that would be optimal if there were no other jobs in the system. That is, AVR processes each job i at the constant speed $w_i/(d_i - r_i)$ throughout interval $[r_i, d_i]$, and the speed of the processor is just the sum of the processing speeds of the jobs. The algorithm OA maintains the invariant that the speed at each time is optimal given the current state, and under the assumption that no more jobs will arrive in the future. In particular, let $w(x)$ denote the amount of unfinished work that has deadline within x time units from the current time. Then the current speed of OA is $\max_x w(x)/x$, this is precisely the speed that the offline optimum algorithm [16] would set in this state. Another online algorithm BKP is proposed in [6]. BKP runs at speed $e \cdot v(t)$ at time t, where

$$v(t) = \max_{t'>t} \frac{w(t, \mathrm{e}t - (\mathrm{e}-1)t', t')}{\mathrm{e}\,(t'-t)}$$

and $w(t, t_1, t_2)$ is the amount of work that has release time at least t_1, deadline at most t_2, and that has already arrived by time t. Clearly, if $w(t_1, t_2)$ is the total work of jobs that are released after t_1 and have deadline before t_2, then any algorithm must have an average speed of at least $w(t_1, t_2)/(t_2 - t_1)$ during

$[t_1, t_2]$. Thus BKP can be viewed as computing a lower bound on the average speed in an online manner and running at e times that speed.

Previous Results

Algorithm	General α		$\alpha = 2$		$\alpha = 3$	
	Upper	Lower	Upper	Lower	Upper	Lower
General		$(4/3)^\alpha/2$		1.1		1.2
AVR	$2^{\alpha-1}\alpha^\alpha$	$(2-\delta)^{\alpha-1}\alpha^\alpha$	8	4	108	48.2
OA	α^α	α^α	4	4	27	27
BKP	$2(\alpha/(\alpha-1))^\alpha e^\alpha$		59.1		135.6	

Our Contributions

Algorithm	General α		$\alpha = 2$		$\alpha = 3$	
	Upper	Lower	Upper	Lower	Upper	Lower
General		$e^{\alpha-1}/\alpha$		1.3		2.4
qOA	$4^\alpha/(2e^{1/2}\alpha^{1/4})$	$(4^\alpha/(4\alpha))(1-2/\alpha)^{\alpha/2}$	2.4		6.7	

Table 1: Results on the competitive ratio for energy minimization with deadline feasibility.

Table 1 summarizes the results in the literature related to the competitive ratio of online algorithms for this problem. The competitive ratio of AVR is at most $2^{\alpha-1}\alpha^\alpha$. This was first shown in [16], and a simpler potential function based analysis was given in [3]. This result is almost tight. In particular, the competitive ratio of AVR is least $(2-\delta)^{\alpha-1}\alpha^\alpha$, where δ is a function of α that approaches zero as α approaches infinity [3]. The competitive ratio of OA is exactly α^α [6], where the upper bound is proved using an amortized local competitiveness argument. Thus the competitive ratio of AVR is strictly inferior to that of OA. The competitive ratio of BKP is at most $2(\alpha/(\alpha-1))^\alpha e^\alpha$ [6], which is about $2e^{\alpha+1}$ for large α. This bound on the competitive ratio of BKP is better than that of OA only for $\alpha \geq 5$. On the other hand, the known lower bounds for general algorithms are rather weak and are based on instances consisting of just two jobs. In particular, [5] show a lower bound of $(4/3)^\alpha/2$ for any deterministic algorithm. If one tries to find the worst 3, 4, ... job instances, the calculations get messy quickly.

The most interesting value of α is certainly 3 as in current CMOS based processors the dynamic power is approximately the cube of the speed (this is commonly called the cube-root rule) [8]. It seems likely that α would be in the range $[2,3]$ for most conceivable devices. The best known guarantee for α in this range is α^α achieved by OA, which evaluates to 4 for $\alpha = 2$ and 27 for $\alpha = 3$.

1.2 Our contributions

In Section 3 we introduce and analyze a natural class of speed scaling algorithms, that we call qOA. The algorithm qOA sets the speed of the processor to be $q \geq 1$ times the speed that the optimal offline algorithm would run the jobs in the current state, or equivalently q times the speed that the algorithm OA would run in the current state. In the worst-case instances for OA the rate that work arrives increases with time. Intuitively, the mistake that the algorithm OA makes in these instances is that it runs too slowly initially as it doesn't anticipate future arrivals. So the motivation of the definition of qOA is to avoid making the same mistake as OA, and to run faster in anticipation of further work arriving in the future.

We show, using an amortized local competitiveness analysis, that if q is set to $2-1/\alpha$, then the competitive ratio of qOA is

$$\left(2-\frac{1}{\alpha}\right)^{\alpha}\left(1+\alpha^{-1/(\alpha-1)}\right)^{\alpha-1},$$

which is at most $4^{\alpha}/(2\mathrm{e}^{1/2}\alpha^{1/4})$. This bound is approximately 3.38 when $\alpha=2$, and 11.52 when $\alpha=3$. Setting $q=2-1/\alpha$ is not necessarily the optimum value of q for our analysis (although it isn't too far off). For general α, it is not clear how to obtain the optimum choice of q for our analysis since this involves solving a system of high degree algebraic inequalities. For the case of $\alpha=3$ and that of $\alpha=2$, we can explicitly determine the the choice of q that gives best bound on the competitive ratios using our analysis. We show that qOA is at worst 2.4-competitive when $\alpha=2$, and at worst 6.7-competitive when $\alpha=3$.

There are two main technical ideas in the competitive analysis of qOA. The first is the introduction of a new potential function which is quite different from the one used in the analysis of OA in [6] and the potential function used to analyze AVR in [3]. The second idea is to use a convexity based argument in the analysis, instead of Young's inequality. The analysis in [7], and almost all of the amortized local competitiveness analyses in the speed scaling literature, rely critically on the Young's inequality. However, in the current setting, Young's inequality gives a bound that is too weak to be useful when analyzing qOA. Instead we observe that certain expressions that arise in the analysis are convex, which allows us to reduce the analysis of the general case down to just two extreme cases.

In Section 4 we give the first non-trivial lower bound on the competitive ratio for a general deterministic algorithm. We show no deterministic algorithm can have a competitive ratio less than $\mathrm{e}^{\alpha-1}/\alpha$. Our lower bound is almost optimal since BKP achieves a ratio of about $2\mathrm{e}^{\alpha+1}$, in particular, the base of the power, e, is the best possible. For $\alpha=3$, this raises the best known lower bound a modest amount, from 1.2 to 2.4.

Given the general lower bound of $\mathrm{e}^{\alpha-1}/\alpha$, and that BKP achieves a ratio with a base of the power e, a natural question is whether there is some choice of the parameter q for which the competitive ratio of qOA varies with e as the base of the power. Somewhat surprisingly, we show that this is not the case and the base of the power cannot be improved beyond 4. In particular, in Section 5 we show that the competitive ratio of qOA can not be better than

$$\frac{4^{\alpha}}{4\alpha}\left(1-\frac{2}{\alpha}\right)^{\alpha/2}.$$

For large α this is about $4^{\alpha-1}/(\alpha e)$. Note that this lower bound essentially matches our upper bound of

$$\frac{4^{\alpha}}{2\mathrm{e}^{1/2}\alpha^{1/4}}$$

for qOA.

Our results are summarized in the last two rows of Table 1. In particular we improve the competitive ratio in the case that the cube-root rule holds from $[1.2,27]$ to $[2.4,6.7]$ and in the case that $\alpha=2$ from $[1.1,4]$ to $[1.3,2.4]$.

1.3 Other related results

There are now enough speed scaling papers in the literature that it is not practical to survey all such papers here. We limit ourselves to those papers most related to the results presented here. Surveys of the speed scaling literature include [1, 10].

A naive implementation of the offline optimum algorithm for deadline feasibility YDS [16] runs in time $O(n^3)$. Faster implementations for discrete and continuous speeds can be found in [11, 13, 14]. [2] considered the problem of finding energy-efficient deadline-feasible schedules on multiprocessors. [2] showed that the offline problem is NP-hard, and gave $O(1)$-approximation algorithms. [2] also gave online algorithms that are $O(1)$-competitive when job deadlines occur in the same order as their release times. [4] investigated speed scaling for deadline feasibility in devices with a regenerative energy source such as a solar cell.

2 Formal problem statement

A problem instance consists of n jobs. Job i has a release time r_i, a deadline $d_i > r_i$, and work $w_i > 0$. In the online version of the problem, the scheduler learns about a job only at its release time; at this time, the scheduler also learns the work and the deadline of the job. We assume that time is continuous. A schedule specifies for each time a job to be run and a speed at which to run the job. The speed is the amount of work performed on the job per unit time. A job with work w run at a constant speed s thus takes w/s units of time to complete. More generally, the work done on a job during a time period is the integral over that time period of the speed at which the job is run. A job i is completed by d_i if work at least w_i is done on it during $[r_i, d_i]$. A schedule is *feasible* if every job is completed by its deadline. Note that the times at which work is performed on job i do not have to be contiguous, that is, preemption is allowed. If the processor is running at speed s, then the power is $P(s) = s^\alpha$ for some constant $\alpha > 1$. The energy used during a time period is the integral of the power over that time period. Our objective is to minimize the total energy subject to completing all jobs by their respective deadlines. An algorithm A is said to be c-competitive if for any instance, the energy usage by A is at most c times that of the optimal schedule. If S is a schedule then we use E_S to denote the energy used by that schedule. If A is an algorithm, and we are considering a fixed instance, then we use E_A to denote the energy used by the schedule produced by A on the instance.

3 Upper bound analysis of qOA

Our goal in this section is to prove the following three theorems.

Theorem 3.1. *When $q = 2 - 1/\alpha$, qOA achieves the competitive ratio*

$$\left(2 - \frac{1}{\alpha}\right)^{\alpha} \left(1 + \alpha^{-1/(\alpha-1)}\right)^{\alpha-1}$$

for general $\alpha > 1$. In particular, this implies that qOA is $4^\alpha/(2\mathrm{e}^{1/2}\alpha^{1/4})$-competitive.

Theorem 3.2. *If $q = 1.54$, then qOA is 6.73-competitive for $\alpha = 3$.*

Theorem 3.3. *If $q = 1.46$, then qOA is 2.39-competitive for $\alpha = 2$.*

We essentially prove Theorems 3.1, 3.2 and 3.3 in parallel; the proofs differ only at the end. We use an amortized local competitiveness analysis, and use a potential function Φ. Although our presentation here should be self-contained, for further background information on amortized local competitiveness arguments see [15]. In this setting, the units of Φ will be energy, and thus, the derivative of Φ with respect to time will be power. Intuitively, Φ is a bank/battery of energy that qOA has saved (over some optimum solution) from the past, that it can use in the future if it lags behind the optimum.

Before defining the potential function Φ, we need to introduce some notations. We always denote the current time as t_0. Since all of our quantities are defined with respect to the current time, we will drop t_0 for notational ease (unless there is cause for confusion). Let s_a and s_o be the current speed of qOA and the optimal algorithm OPT respectively. For any $t_0 \le t' \le t''$, let $w_a(t',t'')$ denote the total amount of unfinished work for qOA at t_0 that has a deadline during $(t',t'']$. Define $w_o(t',t'')$ similarly for OPT. Using this notation, recall that qOA runs at speed

$$s_a = q \cdot \max_{t > t_0} \frac{w_a(t_0,t)}{t - t_0},$$

where $q \ge 1$ will be some fixed constant depending on α.

Let $d(t',t'') = \max\{0, w_a(t',t'') - w_o(t',t'')\}$ denote the excess unfinished work that qOA has relative to OPT among the already released jobs with deadlines in the range $(t',t'']$. We define a sequence of *critical times* $t_0 < t_1 < t_2 < \cdots < t_h$ iteratively as follows: Let t_1 be the latest time such that $d(t_0,t_1)/(t_1 - t_0)$ is maximized. Clearly, t_1 is no more than the latest deadline of any job released thus far. If t_i is earlier than the latest deadline, let $t_{i+1} > t_i$ be the latest time, not later than the latest deadline, that maximizes $d(t_i,t_{i+1})/(t_{i+1} - t_i)$.

We will refer to the intervals $[t_i,t_{i+1}]$ as *critical intervals*. We use g_i to denote $d(t_i,t_{i+1})/(t_{i+1} - t_i)$, which is the density of the excess work with deadline in $(t_i,t_{i+1}]$. We note that $g_0, g_1, \ldots, g_{h-1}$ is a non-negative strictly decreasing sequence. To see this, suppose for the sake of contradiction that this does not hold, and let i be smallest index such that $g_i \ge g_{i-1}$. Then this implies that

$$\frac{d(t_i,t_{i+1})}{t_{i+1} - t_i} \ge \frac{d(t_{i-1},t_i)}{t_i - t_{i-1}} \quad \text{and hence} \quad \frac{d(t_{i-1},t_{i+1})}{t_{i+1} - t_{i-1}} \ge \frac{d(t_{i-1},t_i)}{t_i - t_{i-1}},$$

contradicting the choice of t_i in our iterative procedure. Finally, we note that the quantities t_i and g_i depend on the current time t_0 and might change over time.

We define the potential function Φ as

$$\Phi = \beta \sum_{i=0}^{h-1} (t_{i+1} - t_i) \cdot g_i^{\alpha}$$

where β is some constant which we will optimize later.

We first note some simple observations about Φ, t_i and g_i.

Observation 3.4. Φ is zero before any jobs are released, and after all jobs are completed.

Proof. This directly follows as each $g_i = 0$ by definition. $\square$

Observation 3.5. Job arrivals do not increase Φ, or change the definition of critical times. Also, job completions by either qOA or OPT do not change Φ.

Proof. Upon a job arrival, the work of both online and offline increases exactly by the same amount, and hence the excess work $d(t',t'')$ does not change for any t' and t''. For job completions, we note that $d(t',t'')$ and Φ are both a continuous function of the unfinished work, and the unfinished work on a job continuously decreases to 0 as it completes. $\square$

The critical times thus only change due to qOA or OPT working on the jobs. However, as we show next, this does not cause any discontinuous change in Φ.

Observation 3.6. The instantaneous change in critical times does not (abruptly) change the value of Φ.

Proof. There are three ways the critical times can change.

1. *Merging of two critical intervals:* As qOA follows EDF it must work on jobs with deadline in $[t_0,t_1]$, causing g_0 to decrease until it becomes equal to g_1. At this point, the critical intervals $[t_0,t_1]$ and $[t_1,t_2]$ merge together. Now, Φ does not change by this merger as $g_0 = g_1$ at this point.

2. *Splitting of a critical interval:* As OPT works on some job with deadline $t' \in (t_k,t_{k+1}]$, the quantity
$$\frac{w_a(t_k,t') - w_o(t_k,t')}{t'-t_k}$$
may increase faster than
$$\frac{w_a(t_k,t_{k+1}) - w_o(t_k,t_{k+1})}{t_{k+1}-t_k}$$
causing this interval to split into two critical intervals, $[t_k,t']$ and $[t',t_{k+1}]$. This split does not change Φ as the density of the excess work for both of these newly formed intervals is g_k.

3. *Formation of a new critical time:* A job arrives with later deadline than any previous job, and a new critical time t_{h+1} is created. The potential Φ does not change because $g_h = 0$.

$\square$

The observations above imply that the potential function does not change due to any discrete events such as arrivals, job completions, or changes in critical intervals. Then, in order to establish that qOA is c-competitive with respect to energy, it is sufficient to show the following *running condition* at all times when there is no discrete change as discussed above:

$$s_a^\alpha + \frac{d\Phi}{dt} \le c \cdot s_o^\alpha \,. \tag{3.1}$$

The fact that the running condition holds for all times establishes c-competitiveness follows by integrating the running condition over time, and from the fact that Φ is initially and finally 0, and the fact that Φ does not increase due to discrete events.

In the next three lemmas, we provide simple bounds on the speed s_a of qOA and the speed s_o for OPT, which will be useful in this analysis.

Lemma 3.7. *Without loss of generality, we may assume that* $s_o \geq \max_{t>t_0} \frac{w_o(t_0,t)}{t-t_0}$.

Proof. OPT needs to complete at least $w_o(t_0,t)$ units of work by time t. As the function s^α is convex, the energy optimal way to accomplish this is to run at a constant speed of $w_o(t_0,t)/(t-t_0)$ during $[t_0,t]$. Since OPT is optimum, it may run only faster (due to possible more jobs arriving in the future). □

Lemma 3.8. $s_a \geq qg_0$.

Proof. By definition of qOA, we have that,

$$s_a = q \cdot \max_{t>t_0} \frac{w_a(t_0,t)}{t-t_0} \geq q \cdot \frac{w_a(t_0,t_1)}{t_1-t_0} \geq q \cdot \frac{d(t_0,t_1)}{t_1-t_0} = qg_0 .$$

□

Lemma 3.9. $s_a \leq qg_0 + qs_o$.

Proof. By the definition of qOA and $d(t_0,t)$, we have that,

$$\begin{aligned} s_a = q \cdot \max_{t>t_0} \frac{w_a(t_0,t)}{t-t_0} &\leq q \cdot \max_{t>t_0} \frac{w_o(t_0,t)+d(t_0,t)}{t-t_0} \\ &\leq q \cdot \max_{t>t_0} \frac{w_o(t_0,t)}{t-t_0} + q \cdot \max_{t>t_0} \frac{d(t_0,t)}{t-t_0} \\ &\leq qs_o + qg_0 . \end{aligned}$$

Here the last inequality follows by Lemma 3.7 and the defintion of g_0. □

We are now ready to prove Theorems 3.1, 3.2 and 3.3. Let us first consider the easy case when $w_a(t_0,t_1) \leq w_o(t_0,t_1)$.

Case 1: Suppose that $w_a(t_0,t_1) \leq w_o(t_0,t_1)$. Now by definition, $d(t_0,t_1) = 0$ and $g_0 = 0$, and hence $s_a \leq qs_o$ by Lemma 3.9. Note that there is only one critical interval $[t_0,t_1]$ and

$$\frac{d\Phi}{dt_0} = \frac{d}{dt_0}(\beta(t_1-t_0)\cdot g_0^\alpha) = \beta(t_1-t_0)\cdot \alpha g_0^{\alpha-1}\frac{dg_0}{dt_0} - \beta \cdot g_0^\alpha = 0 .$$

Thus to show (3.1) it suffices to show that $q^\alpha \leq c$, which is easily verified for our choice of q and c in each of Theorems 3.1, 3.2 and 3.3.

Case 2: Henceforth we assume that $w_a(t_0,t_1) > w_o(t_0,t_1)$. Note that $g_0 > 0$ in this case. As qOA follows EDF, it must work on some job with deadline at most t_1, and hence $w_a(t_0,t_1)$ decreases at rate s_a. For OPT, let s_o^k denote the speed with which OPT works on jobs with deadline in the critical interval $(t_k,t_{k+1}]$. We need to determine $d\Phi/dt_0$. To this end, we make the following observation.

Observation 3.10. For $k > 0$, g_k increases at rate at most $s_o^k/(t_{k+1}-t_k)$. For $k = 0$, g_0 changes at rate

$$\frac{-s_a + s_o^0}{t_1-t_0} + \frac{g_0}{t_1-t_0} .$$

Proof. Let us first consider $k>0$. The rate of change $t_{k+1}-t_k$ with respect to t_0 is 0. Moreover, as qOA does not work on jobs in $w_a(t_k,t_{k+1})$ and $w_o(t_k,t_{k+1})$ decreases at rate s_o^k, it follows that

$$\frac{d}{dt_0}g_k = \frac{d}{dt_0}\frac{\max(0, w_a(t_k,t_{k+1}) - w_o(t_k,t_{k+1}))}{t_{k+1}-t_k} \le \frac{s_o^k}{t_{k+1}-t_k}\,.$$

For $k=0$, and by our assumption that $w_a(t_0,t_1) > w_o(t_0,t_1)$, we have that

$$\frac{d}{dt_0}g_0 = \frac{d}{dt_0}\frac{w_a(t_0,t_1)-w_o(t_0,t_1)}{t_1-t_0} = \frac{-s_a+s_o^0}{t_1-t_0} + \frac{w_a(t_0,t_1)-w_o(t_0,t_1)}{(t_1-t_0)^2} = \frac{-s_a+s_o^0}{t_1-t_0} + \frac{g_0}{t_1-t_0}\,.$$

□

Now,

$$\begin{aligned}
\frac{d\Phi}{dt_0} &= \beta\left(\frac{d}{dt_0}((t_1-t_0)\cdot g_0^\alpha) + \sum_{k>0}\frac{d}{dt_0}((t_{k+1}-t_k)\cdot g_k^\alpha)\right)\\
&= \beta\left((t_1-t_0)\cdot \alpha g_0^{\alpha-1}\frac{dg_0}{dt_0} - g_0^\alpha + \sum_{k>0}\alpha(t_{k+1}-t_k)g_k^{\alpha-1}\frac{dg_k}{dt_0}\right)\\
&\le \beta\left((-s_a+s_o^0)\alpha g_0^{\alpha-1} + (\alpha-1)g_0^\alpha + \sum_{k>0}\alpha s_o^k g_k^\alpha - 1\right) \qquad (3.2)
\end{aligned}$$

where the last step follows from the second step using Observation 3.10.

As the g_i's are non-increasing and $\sum_{k\ge 0}s_o^k = s_o$ by definition, (3.2) implies that

$$\frac{d\Phi}{dt} \le \beta(\alpha g_0^{\alpha-1}(-s_a+s_o) + (\alpha-1)g_0^\alpha)\,.$$

Thus to show the running condition (3.1), it is sufficient to show that

$$s_a^\alpha + \beta(\alpha g_0^{\alpha-1}(-s_a+s_o) + (\alpha-1)g_0^\alpha) - c\cdot s_o^\alpha \le 0\,. \qquad (3.3)$$

Consider the left hand side of equation (3.3) as a function of s_a while g_0 and s_o are fixed. We note that it is a convex function of s_a. Hence, to show (3.3), it is sufficient to show that it holds at the extreme possible values for s_a, which by Lemma 3.8 and Lemma 3.9 are $s_a = qg_0$ and $s_a = qg_0 + qs_o$.

For $s_a = qg_0$, the left hand side of (3.3) becomes

$$(q^\alpha - \beta\alpha q + \beta(\alpha-1))g_0^\alpha + \beta\alpha g_0^{\alpha-1}s_o - cs_o^\alpha\,. \qquad (3.4)$$

Taking derivative with respect to s_o, we see that this is maximized when $cs_o^{\alpha-1} = \beta g_0^{\alpha-1}$. Substituting this value for s_o and canceling g_0^α on both sides, it follows that it suffices to satisfy:

$$(q^\alpha - \beta\alpha q + \beta(\alpha-1)) + \beta(\alpha-1)\left(\frac{\beta}{c}\right)^{1/(\alpha-1)} \le 0\,. \qquad (3.5)$$

Next, for $s_a = qg_0 + qs_o$, the left hand side of equation (3.3) becomes

$$q^\alpha(g_0+s_o)^\alpha - \beta(q\alpha-(\alpha-1))g_0^\alpha - \beta\alpha(q-1)g_0^{\alpha-1}s_o - cs_o^\alpha\,. \tag{3.6}$$

Substituting $s_o = x\cdot g_0$ and canceling g_0^α on both sides, it suffices to satisfy

$$q^\alpha(1+x)^\alpha - \beta(q\alpha-(\alpha-1)) - \beta\alpha(q-1)x - cx^\alpha \le 0\,. \tag{3.7}$$

We now fork our proofs of Theorems 3.1, 3.2 and 3.3. In each case we need to show that equations (3.5) and (3.7) hold. We first finish up the proof for Theorem 3.1. Recall that we set $q = 2-1/\alpha$. We let $\beta = c = q^\alpha\eta^{\alpha-1}$ where $\eta = 1+\alpha^{-1/(\alpha-1)}$. With these choices of q,β and c, $\alpha q = 2\alpha-1$. Substituting in equation (3.5), and dividing through by q^α, we obtain that this equation is then equivalent to

$$\left(1-\eta^{\alpha-1}(2\alpha-1)+\eta^{\alpha-1}(\alpha-1)\right)+\eta^{\alpha-1}(\alpha-1)\le 0$$

which is equivalent to $\eta \ge 1$. Similarly, equation (3.7) is equivalent to

$$(1+x)^\alpha - \alpha\eta^{\alpha-1} - \eta^{\alpha-1}(\alpha-1)x - \eta^{\alpha-1}x^\alpha \le 0\,.$$

Since $\alpha \ge 1$, it suffices to show that

$$(1+x)^\alpha - \alpha\eta^{\alpha-1} - \eta^{\alpha-1}x^\alpha \le 0\,. \tag{3.8}$$

Taking the derivative with respect to x of the left side of equation (3.8), we can conclude that the maximum is attained at x such that $(1+x)^{\alpha-1} - \eta^{\alpha-1}x^{\alpha-1} = 0$, or equivalently $x = 1/(\eta-1) = \alpha^{1/(\alpha-1)}$. For this value of x, the left side of equation (3.8) evaluates to 0 and hence the result follows. Hence the running condition (3.3) is satisfied. So the competitive ratio is at most $c = q^\alpha\eta^{\alpha-1}$ with $q = 2-1/\alpha$ and $\eta = 1+\alpha^{-1/(\alpha-1)}$, which implies Theorem 3.1 holds.

To obtain the bound $4^\alpha/(2\mathrm{e}^{1/2}\alpha^{1/4})$, we note that $(1-1/x)^x \le 1/\mathrm{e}$ for $x>1$ and hence

$$q^\alpha = \left(2\left(1-\frac{1}{2\alpha}\right)\right)^\alpha \le 2^\alpha/\sqrt{\mathrm{e}}\,.$$

Similarly, as $\mathrm{e}^{-x} \le 1-x+x^2/2 \le 1-x/2$ for $0\le x<1$, and $\ln(\alpha)\le(\alpha-1)$ for $\alpha>1$ we have

$$\eta = 1+\alpha^{-1/(\alpha-1)} = 1+\mathrm{e}^{-\ln\alpha/(\alpha-1)} \le 2-\frac{\ln\alpha}{2(\alpha-1)} = 2\left(1-\frac{\ln\alpha}{4(\alpha-1)}\right).$$

Thus

$$\eta^{\alpha-1} = 2^{\alpha-1}\left(1-\frac{\ln\alpha}{4(\alpha-1)}\right)^{\frac{4(\alpha-1)}{\ln\alpha}\cdot\frac{\ln\alpha}{4}} \le 2^{\alpha-1}\mathrm{e}^{-(\ln\alpha)/4} = 2^{\alpha-1}/\alpha^{1/4}$$

which implies the overall bound.

To finish the proof of Theorem 3.2 we wish to determine the values of q and β so that the inequalities (3.5) and (3.7) hold with the minimum possible value of c. Plugging $\alpha=3$ into inequalities (3.5) and (3.7) we obtain:

$$\begin{aligned}(q^3-3\beta q+2\beta)+2\beta\left(\frac{\beta}{c}\right)^{1/2} &\le 0\,, \quad \text{and}\\ q^3(1+x)^3-\beta(3q-2)-3\beta(q-1)x-cx^3 &\le 0\,.\end{aligned}$$

We wrote a computer program to approximately determine the values of q and β that minimize c. The best values we obtained are $q = 1.54, \beta = 7.78$ and $c = 6.73$. It is easy to check that (3.5) is satisfied. The left hand side of (3.7) becomes $-3.08x^3 + 10.96x^2 - 1.65x - 16.73$, which can be shown to be negative by differentiation. A similar process yields Theorem 3.3. We set $q = 1.46$ and $\beta = 2.7$ and check that (3.5) and (3.7) are satisfied for these values to give $c = 2.39$.

4 General lower bound

The goal in this section is to prove the following theorem.

Theorem 4.1. *No deterministic online algorithm A can have a competitive ratio less than* $\mathrm{e}^{\alpha-1}/\alpha$.

We assume α is fixed and is known to the algorithm. We give an adversarial strategy for constructing a job instance based on the behavior of A. We demonstrate a schedule OPT whose energy usage is arbitrarily close to a factor of $\mathrm{e}^{\alpha-1}/\alpha$ less than the energy used by the schedule produced by A.

Adversarial strategy: Let $\varepsilon > 0$ be some small fixed constant. Work is arriving during $[0,h]$, where $0 < h \leq 1-\varepsilon$. The rate of work arriving at time $t \in [0,h]$ is

$$a(t) = \frac{1}{1-t}\,.$$

So the work that arrives during any time interval $[u,v]$ is $\int_u^v a(t)dt$. All work has deadline 1. The value of h will be set by the adversary according to the action of A. Intuitively, if A spends too much energy initially, then h will be set to be small. If A does not spend enough energy early on, then h will be set to $1-\varepsilon$. In this case, A will have a lot of work left toward the end and will have to spend too much energy finishing this work off. To make this more formal, consider the function

$$E(t) = \int_0^t \left(\left(1+\frac{b}{\ln\varepsilon}\right)\frac{1}{1-x}\right)^{\alpha} dx$$

where b is some constant (which we will later set to $1/(\alpha-1)^{1/\alpha}$). This is the total energy usage up to time t if A runs at speed

$$s(t) = \left(1+\frac{b}{\ln\varepsilon}\right)\frac{1}{1-t}\,.$$

Of course, A may run at speed other than $s(t)$. We set h be the first time, satisfying $0 < h < 1-\varepsilon$, such that total energy usage of A up to time h is at least $E(h)$. If no such time exists, then $h = 1-\varepsilon$.

We break the lower bounding of the competitive ratio into three cases depending on whether $h \in (0, 1-1/\mathrm{e}]$, $h \in (1-1/\mathrm{e}, 1-\varepsilon)$, or $h = 1-\varepsilon$. The proofs of some inequalities are given in lemmas at the end of the section.

The case that $h \in (0, 1-1/\mathrm{e}]$: Since $h < 1-\varepsilon$, the energy used by the algorithm E_A is at least $E(h)$ and hence

$$
\begin{aligned}
E_A \ \geq\ E(h) &= \int_0^h \left(\left(1+\frac{b}{\ln\varepsilon}\right)\frac{1}{1-x}\right)^{\alpha} dx \\
&= \left(1+\frac{b}{\ln\varepsilon}\right)^{\alpha}\left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}}-\frac{1}{\alpha-1}\right). \qquad (4.1)
\end{aligned}
$$

Now consider the possible schedule OPT that runs at a constant speed $s_o = \ln(1/(1-h))$ throughout $[0,1]$. Since $h \leq 1-1/\mathrm{e}$, we have $s_o \leq 1$. On the other hand, $a(t) \geq 1$ for all $t \in [0,h]$. Hence, during $[0,h]$, there is always enough released work for OPT to process. Observe that the total processing done by OPT in $[0,1]$ is $\ln(1/(1-h))$, which equals the total work released, so OPT is feasible.

We can calculate a bound on the energy used by OPT as follows:

$$
\begin{aligned}
E_{\mathrm{OPT}} = s_o^{\alpha} &= \left(\ln\frac{1}{1-h}\right)^{\alpha} \\
&\leq \frac{\alpha}{\mathrm{e}^{\alpha-1}}\left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}}-\frac{1}{\alpha-1}\right)
\end{aligned}
$$

where the inequality is proved in Lemma 4.2.

Combining our bounds on E_A and E_{OPT}, we can conclude that the competitive ratio in this case is at least

$$\left(1+\frac{b}{\ln\varepsilon}\right)^{\alpha}\frac{1}{\alpha}\mathrm{e}^{\alpha-1}$$

which tends to $\mathrm{e}^{\alpha-1}/\alpha$ as ε tends to 0.

The case that $h \in (1-1/\mathrm{e}, 1-\varepsilon)$: One possible schedule OPT is to run at speed $s_o(t) = a(t)$ for $t \in [0, 1-\mathrm{e}(1-h)]$ and run at a constant speed $s_o(t) = 1/(\mathrm{e}(1-h))$ for $t \in [1-\mathrm{e}(1-h), 1]$. Note that by simple algebra, we have $0 < 1-\mathrm{e}(1-h) < h$. We observe that $s_o(t) \leq a(t)$ for all $t \in [0,h]$, hence there is always enough released work for OPT to process during $[0,h]$. To establish that OPT is feasible, we show that the total processing done by OPT equals the total work released, as follows.

$$
\begin{aligned}
\int_0^1 s_o(t)dt &= \int_0^{1-\mathrm{e}(1-h)} a(t)dt + \int_{1-\mathrm{e}(1-h)}^1 \frac{1}{\mathrm{e}(1-h)}dt \\
&= \left(\ln\frac{1}{\mathrm{e}(1-h)}\right) + \left(\frac{(1-(1-\mathrm{e}(1-h)))}{\mathrm{e}(1-h)}\right) \\
&= \ln\frac{1}{1-h} \\
&= \int_0^h a(t)dt\,.
\end{aligned}
$$

We now wish to bound the energy used by OPT.

$$\begin{aligned}
E_{\mathrm{OPT}} &= \int_0^1 (s_o(t))^\alpha dt \\
&= \int_0^{1-\mathrm{e}(1-h)} \left(\frac{1}{1-t}\right)^\alpha dt + \int_{1-\mathrm{e}(1-h)}^1 \left(\frac{1}{\mathrm{e}(1-h)}\right)^\alpha dt \\
&= \frac{1}{(\alpha-1)\mathrm{e}^{\alpha-1}(1-h)^{\alpha-1}} - \frac{1}{\alpha-1} + \left(\frac{1}{\mathrm{e}(1-h)}\right)^{\alpha-1} \\
&= \frac{\alpha}{\mathrm{e}^{\alpha-1}} \frac{1}{(\alpha-1)(1-h)^{\alpha-1}} - \frac{1}{\alpha-1}\,.
\end{aligned} \tag{4.2}$$

By the fact that $\mathrm{e}^x \geq 1+x$ for all $x \geq 0$, we have that $\mathrm{e}^{\alpha-1} \geq \alpha$ and $\alpha/\mathrm{e}^{\alpha-1} \leq 1$. Hence, we can loosen the above bound to:

$$E_{\mathrm{OPT}} \leq \frac{\alpha}{\mathrm{e}^{\alpha-1}} \left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}} - \frac{1}{\alpha-1}\right).$$

The following bound on E_A from line (4.1) still holds in this case:

$$E_A \geq \left(1+\frac{b}{\ln\varepsilon}\right)^\alpha \left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}} - \frac{1}{\alpha-1}\right).$$

Combining the bounds on E_A and E_{OPT}, we again conclude that the competitive ratio is at least

$$\left(1+\frac{b}{\ln\varepsilon}\right)^\alpha \frac{1}{\alpha}\mathrm{e}^{\alpha-1}$$

which tends to $\mathrm{e}^{\alpha-1}/\alpha$ as ε tends to 0.

The case that $h = 1-\varepsilon$: Note that the adversary ends the arrival of work at time $1-\varepsilon$ and the total amount of work arrived is

$$\int_0^{1-\varepsilon} \frac{1}{1-t} dt = -\ln\varepsilon\,.$$

Also note that the total energy usage of A up to $1-\varepsilon$ may be exactly $E(1-\varepsilon)$.

We first show that much of the work released is unfinished by A at time $1-\varepsilon$. To see this let $s_A(t)$ be the speed of the algorithm A at time t and consider the algorithm B that works at speed

$$s_B(t) = \left(1+\frac{b}{\ln\varepsilon}\right)\frac{1}{1-t}\,.$$

The energy consumed by B by time $t \leq 1+\varepsilon$ is exactly

$$\int_0^t s_B(x)^\alpha dx = E(t)\,,$$

which, by the definition of h, is at least the energy consumed by A by time t. We will prove in Lemma 4.3 that this implies the work processed by A by time $1-\varepsilon$ is at most the work processed by B by time $1-\varepsilon$. Hence the maximum amount of work completed by A by time $1-\varepsilon$ is

$$\begin{aligned}\int_0^{1-\varepsilon} s_B(t)dt &= \int_0^{1-\varepsilon}\left(1+\frac{b}{\ln\varepsilon}\right)\frac{1}{1-x}dx\\ &= -\ln\varepsilon - b\,.\end{aligned}$$

Hence, A has at least b units of work remaining at time $1-\varepsilon$. To complete this amount of work during $[1-\varepsilon,1]$,

$$E_A \geq \int_{1-\varepsilon}^{1}(s_A(t))^\alpha dt = \left(\frac{b}{\varepsilon}\right)^\alpha \varepsilon = \frac{1}{(\alpha-1)\varepsilon^{\alpha-1}}$$

where the last equality follows from setting $b = 1/(\alpha-1)^{1/\alpha}$. Using the bound on E_{OPT} from line (4.2), with $h = 1-\varepsilon$, we find that there is a feasible schedule using energy at most

$$\frac{\alpha}{\mathrm{e}^{\alpha-1}}\frac{1}{(\alpha-1)\varepsilon^{\alpha-1}}\,.$$

Thus, the competitive ratio in this case is at least $\mathrm{e}^{\alpha-1}/\alpha$.

To finish the proof of Theorem 4.1 we need the following two technical lemmas.

Lemma 4.2. *For any $h \in (0, 1-1/\mathrm{e}]$,*

$$\left(\ln\frac{1}{1-h}\right)^\alpha \leq \frac{\alpha}{\mathrm{e}^{\alpha-1}}\left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}} - \frac{1}{\alpha-1}\right).$$

Proof. Let us define

$$f(h) = \left(\ln\frac{1}{1-h}\right)^\alpha - \frac{\alpha}{\mathrm{e}^{\alpha-1}}\left(\frac{1}{(\alpha-1)(1-h)^{\alpha-1}} - \frac{1}{\alpha-1}\right).$$

Differentiating $f(h)$, we have

$$\begin{aligned}f'(h) &= \alpha\left(\ln\frac{1}{1-h}\right)^{\alpha-1}\frac{1}{1-h} - \frac{\alpha}{\mathrm{e}^{\alpha-1}}\frac{1}{(1-h)^\alpha}\\ &= \frac{\alpha}{1-h}\left(\left(\ln\frac{1}{1-h}\right)^{\alpha-1} - \left(\frac{1}{\mathrm{e}(1-h)}\right)^{\alpha-1}\right).\end{aligned}$$

We can check easily by differentiation that

$$\ln\frac{1}{1-h} \leq \frac{1}{\mathrm{e}(1-h)}$$

for all $h > 0$, and the equality holds only at $h = 1-1/\mathrm{e}$. Therefore, $f'(h)$ is non-positive, and $f(h) \leq f(0) = 0$. The Lemma then follows. □

Lemma 4.3. *Let $s_A(t)$ and $s_B(t)$ be non-negative speed functions for two algorithms A and B over a time interval $[0,x]$. Assume $s_B(t)$ is positive, continuous, differentiable, and monotonically increasing. Then if A always has used no more energy than B, that is if*

$$\int_0^y (s_A(t))^\alpha dt \le \int_0^y (s_B(t))^\alpha dt \quad \textit{for all } y \in [0,x]\,,$$

then A has processed no more work than B by time x, that is

$$\int_0^x s_A(t)dt \le \int_0^x s_B(t)dt\,.$$

Proof. For any $y \in [0,x]$, define

$$F(y) = \int_0^y (s_A(t)^\alpha - s_B(t)^\alpha)dt \quad \text{and} \quad G(y) = \alpha \int_0^y s_B(t)^{\alpha-1}(s_A(t) - s_B(t))dt\,.$$

By Bernoulli's Inequality, $(1+z)^\alpha \ge 1+\alpha z$ for all $\alpha > 1$ and $z \in [-1,\infty)$. Hence,

$$\begin{aligned} F(y) &= \int_0^y \left(s_B(t)^\alpha \left((1 + \frac{s_A(t) - s_B(t)}{s_B(t)})^\alpha - 1 \right) \right) dt \\ &\ge \alpha \int_0^y s_B(t)^{\alpha-1}(s_A(t) - s_B(t))dt \;=\; G(y)\,. \end{aligned}$$

Since $F(y) \le 0$, it follows that $G(y) \le 0$. As s_B is monotonically increasing and positive, it follows that $G(y)s'_B(y)/s_B(y)^\alpha \le 0$ for all $y \in [0,x]$. Hence

$$\int_0^x G(y)\frac{s'_B(y)}{s_B(y)^\alpha}dy \le 0\,.$$

Applying integration by parts and noting that $G'(y) = \alpha s_B(y)^{\alpha-1}(s_A(y) - s_B(y))$, we obtain that

$$\begin{aligned} 0 &\ge \int_0^x G(y)\frac{s'_B(y)}{s_B(y)^\alpha}dy \\ &= \left[G(y)\frac{s_B(y)^{1-\alpha}}{1-\alpha} \right]_0^x - \int_0^x \alpha \frac{s_A(y) - s_B(y)}{1-\alpha}dy \\ &= -G(x)\frac{s_B(x)^{1-\alpha}}{\alpha - 1} + \frac{\alpha}{\alpha-1}\int_0^x (s_A(y) - s_B(y))dy\,. \end{aligned}$$

The last equality follows from $G(0) = 0$. Since $G(x) \le 0$, we obtain the desired result. □

5 Lower Bounds for qOA

Our goal in this section is to prove the following theorem.

Theorem 5.1. *Let $\alpha > 2$ be a fixed known constant. For any choice of q, the competitive ratio of qOA is at least*

$$\frac{1}{4\alpha}4^{\alpha}\left(1-\frac{2}{\alpha}\right)^{\alpha/2}.$$

We show that on the following instance, the energy used by qOA is at least $(4^{\alpha}/(4\alpha))(1-2/\alpha)^{\alpha/2}$ times optimal.

Instance Definition: Let ε be some constant satisfying $\varepsilon \in (0,1)$. Consider the input job sequence where work is arriving during $[0,1-\varepsilon]$ and the rate of arrival at time t is

$$a(t) = \frac{1}{(1-t)^{\beta}} \ ,$$

where $\beta = 2/\alpha$. All work has deadline 1. Finally, a job is released at time $1-\varepsilon$ with work $\varepsilon^{1-\beta}$ and deadline 1.

We first give an upper bound on the energy used in the optimal energy schedule. Consider the schedule OPT that runs at speed $a(t)$ during $[0,1-\varepsilon]$ and then runs at speed $1/\varepsilon^{\beta}$ during $[1-\varepsilon,1]$. Clearly OPT completes all work by deadline 1. The energy usage of OPT, and hence an upper bound on the energy used by an optimal schedule, is then

$$\begin{aligned} E_{\text{OPT}} &= \int_0^{1-\varepsilon}(a(t))^{\alpha}dt + \left(\frac{1}{\varepsilon^{\beta}}\right)^{\alpha}\cdot\varepsilon = \int_0^{1-\varepsilon}\frac{1}{(1-t)^{\beta\alpha}}dt + \frac{1}{\varepsilon^{\beta\alpha-1}} \\ &= \left[\frac{1}{\beta\alpha-1}\frac{1}{(1-t)^{\beta\alpha-1}}\right]_0^{1-\varepsilon} + \frac{1}{\varepsilon^{\beta\alpha-1}} = \frac{1}{\beta\alpha-1}\frac{1}{\varepsilon^{\beta\alpha-1}} - \frac{1}{\beta\alpha-1} + \frac{1}{\varepsilon^{\beta\alpha-1}} \\ &\le \frac{\beta\alpha}{\beta\alpha-1}\frac{1}{\varepsilon^{\beta\alpha-1}}\,. \end{aligned} \tag{5.1}$$

We want to calculate the energy usage of qOA during $[1-\varepsilon,1]$, which is a lower bound to E_{qOA}, as follows. Let $s(t)$ be the speed of qOA at time t. We first determine $s(t)$ for $t \in [0,1-\varepsilon]$. The value of $s(t)$ is the unique nonnegative solution to

$$s(t) = q\cdot\frac{\int_0^t a(y)dy - \int_0^t s(y)dy}{1-t}\,.$$

One can verify that

$$s(t) = \frac{q}{\beta+q-1}\frac{1}{(1-t)^{\beta}} - \frac{q}{\beta+q-1}(1-t)^{q-1}$$

satisfies the above equation by substituting it to the right hand side of the equation, as follows:

$$\begin{aligned}
& \frac{q}{1-t}\left(\int_0^t \frac{1}{(1-y)^\beta}dy - \int_0^t \left(\frac{q}{\beta+q-1}\frac{1}{(1-y)^\beta} - \frac{q}{\beta+q-1}(1-y)^{q-1}\right)dy\right) \\
= \;& \frac{q}{1-t}\int_0^t \left((1-\frac{q}{\beta+q-1})\frac{1}{(1-y)^\beta} + \frac{q}{\beta+q-1}(1-y)^{q-1}\right)dy \\
= \;& \frac{q}{1-t}\cdot\left[\frac{1}{\beta+q-1}(1-y)^{-\beta+1} - \frac{1}{\beta+q-1}(1-y)^q\right]_0^t \\
= \;& \frac{q}{\beta+q-1}\frac{1}{(1-t)^\beta} - \frac{q}{\beta+q-1}(1-t)^{q-1} \;=\; s(t)\,.
\end{aligned}$$

We then bound the amount of work w that qOA has unfinished right after the last job arrives at time $1-\varepsilon$ as follows:

$$\begin{aligned}
w \;&=\; \int_0^{1-\varepsilon} a(t)dt - \int_0^{1-\varepsilon} s(t)dt + \varepsilon^{1-\beta} \\
&=\; \int_0^{1-\varepsilon}\left(\left(1-\frac{q}{\beta+q-1}\right)\frac{1}{(1-t)^\beta} + \frac{q}{\beta+q-1}(1-t)^{q-1}\right)dt + \varepsilon^{1-\beta} \\
&=\; \left[\frac{1}{\beta+q-1}(1-t)^{-\beta+1} - \frac{1}{\beta+q-1}(1-t)^q\right]_0^{1-\varepsilon} + \varepsilon^{1-\beta} \\
&=\; \frac{\beta+q}{\beta+q-1}\frac{1}{\varepsilon^{\beta-1}} - \frac{1}{\beta+q-1}\varepsilon^q\,. \qquad (5.2)
\end{aligned}$$

The speed $s(t)$, for $t \in [1-\varepsilon, 1]$, is the unique solution to the equation

$$s(t) = q \cdot \frac{w - \int_{1-\varepsilon}^t s(y)dy}{1-t}\,.$$

By substitution one can verify that the solution is

$$s(t) = q\left(\frac{\varepsilon}{1-t}\right)^{1-q}\left(\frac{w}{\varepsilon}\right).$$

We can check that $\int_{1-\varepsilon}^1 s(t)dt = w$, so qOA just finishes all the work at time 1. Hence, the energy usage of qOA during $[1-\varepsilon, 1]$ is

$$\begin{aligned}
\int_{1-\varepsilon}^1 (s(t))^\alpha dt \;&=\; q^\alpha \varepsilon^{(1-q)\alpha}\left(\frac{w}{\varepsilon}\right)^\alpha \int_{1-\varepsilon}^1 \frac{1}{(1-t)^{(1-q)\alpha}}dt \\
&=\; \frac{q^\alpha}{q\alpha-\alpha+1}\left(\frac{w}{\varepsilon}\right)^\alpha \varepsilon\,. \qquad (5.3)
\end{aligned}$$

As ε approaches 0, we can see from equation (5.2) that w approaches

$$\frac{\beta+q}{\beta+q-1}\frac{1}{\varepsilon^{\beta-1}}\,.$$

Hence, combining this with equation (5.3), the energy used by qOA at least approaches

$$\frac{q^{\alpha}}{q\alpha-\alpha+1}\left(\frac{\beta+q}{\beta+q-1}\right)^{\alpha}\frac{1}{\varepsilon^{\beta\alpha-1}}\,.$$

Combining this with our bound on the optimal energy from equation (5.1), and setting $\beta=2/\alpha$, we can conclude that the competitive ratio of qOA at least approaches

$$\begin{aligned}\frac{q^{\alpha}}{q\alpha-\alpha+1}\left(\frac{\beta+q}{\beta+q-1}\right)^{\alpha}\left(\frac{\beta\alpha-1}{\beta\alpha}\right) &\geq \frac{q^{\alpha}}{q\alpha}\left(\frac{\beta+q}{\beta+q-1}\right)^{\alpha}\left(\frac{\beta\alpha-1}{\beta\alpha}\right)\\ &= \frac{1}{2\alpha}q^{\alpha-1}\left(1+\frac{1}{q+\frac{2}{\alpha}-1}\right)^{\alpha}.\end{aligned}$$

We can see by differentiation that the above expression is increasing when $q\geq 2$. Hence, to obtain a lower bound for the competitive ratio, we can assume $q\leq 2$. We rewrite the lower bound for the competitive ratio as

$$\frac{1}{2q\alpha}\left(\frac{q(q+\frac{2}{\alpha})}{q+\frac{2}{\alpha}-1}\right)^{\alpha} \tag{5.4}$$

and focus on the term

$$\frac{q(q+\frac{2}{\alpha})}{q+\frac{2}{\alpha}-1}\,.$$

We differentiate with respect to q and get

$$\frac{(q+\frac{2}{\alpha}-1)(2q+\frac{2}{\alpha})-(q^2+\frac{2q}{\alpha})}{(q+\frac{2}{\alpha}-1)^2}\,.$$

Setting this equal to 0, we obtain

$$q^2+\left(\frac{4}{\alpha}-2\right)q+\left(\frac{4}{\alpha^2}-\frac{2}{\alpha}\right)=0\,.$$

The unique positive solution for this equation is

$$q=1-\frac{2}{\alpha}+\sqrt{1-\frac{2}{\alpha}}\,.$$

Plugging this value of q back into our lower bound for the competitive ratio in line (5.4), we finally conclude that the competitive ratio of qOA is at least

$$\begin{aligned}&\frac{1}{2q\alpha}\left(\frac{\left(1-\frac{2}{\alpha}+\sqrt{1-\frac{2}{\alpha}}\right)\left(1+\sqrt{1-\frac{2}{\alpha}}\right)}{\sqrt{1-\frac{2}{\alpha}}}\right)^{\alpha} = \frac{1}{2q\alpha}\left(\left(1+\sqrt{1-\frac{2}{\alpha}}\right)^{2}\right)^{\alpha}\\ \geq\ &\frac{1}{2q\alpha}\left(4\sqrt{1-\frac{2}{\alpha}}\right)^{\alpha} = \frac{1}{2q\alpha}4^{\alpha}\left(1-\frac{2}{\alpha}\right)^{\alpha/2}.\end{aligned}$$

(We used the inequality $(x+y)^2\geq 4xy$.) The Theorem follows since $q\leq 2$.

References

[1] SUSANNE ALBERS: Energy-efficient algorithms. *Comm. ACM*, 53(5):86–96, 2010. [doi:10.1145/1735223.1735245] 213

[2] SUSANNE ALBERS, FABIAN MÜLLER, AND SWEN SCHMELZER: Speed scaling on parallel processors. In *Proc. 19th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA'07)*, pp. 289–298. ACM Press, 2007. [doi:10.1145/1248377.1248424] 210, 213

[3] NIKHIL BANSAL, DAVID P. BUNDE, HO-LEUNG CHAN, AND KIRK PRUHS: Average rate speed scaling. In *Proc. 8th Latin Amer. Symp. on Theoretical Informatics (LATIN'08)*, pp. 240–251. Springer, 2008. [ACM:1792939] 210, 211, 212

[4] NIKHIL BANSAL, HO-LEUNG CHAN, AND KIRK PRUHS: Speed scaling with a solar cell. *Theoret. Comput. Sci.*, 410(45):4580–4587, 2009. Preliminary version in AAIM'08. [doi:10.1016/j.tcs.2009.07.004] 213

[5] NIKHIL BANSAL, TRACY KIMBREL, AND KIRK PRUHS: Dynamic speed scaling to manage energy and temperature. In *Proc. 45th FOCS*, pp. 520–529. IEEE Comp. Soc. Press, 2004. [doi:10.1109/FOCS.2004.24] 211

[6] NIKHIL BANSAL, TRACY KIMBREL, AND KIRK PRUHS: Speed scaling to manage energy and temperature. *J. ACM*, 54(1):3:1–3:39, March 2007. [doi:10.1145/1206035.1206038] 210, 211, 212

[7] NIKHIL BANSAL, KIRK PRUHS, AND CLIFFORD STEIN: Speed scaling for weighted flow time. *SIAM J. Comput.*, 39(4):1294–1308, 2009. Preliminary version in SODA'07. [doi:10.1137/08072125X] 212

[8] DAVID M. BROOKS, PRADIP BOSE, STANLEY E. SCHUSTER, HANS JACOBSON, PRABHAKAR N. KUDVA, ALPER BUYUKTOSUNOGLU, JOHN-DAVID WELLMAN, VICTOR ZYUBAN, MANISH GUPTA, AND PETER W. COOK: Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000. [doi:10.1109/40.888701] 211

[9] HO-LEUNG CHAN, WUN-TAT CHAN, TAK-WAH LAM, LAP-KEI LEE, KIN-SUM MAK, AND PRUDENCE W. H. WONG: Energy efficient online deadline scheduling. In *Proc. 18th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'07)*, pp. 795–804. ACM Press, 2007. [ACM:1283468] 210

[10] SANDY IRANI AND KIRK R. PRUHS: Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005. [doi:10.1145/1067309.1067324] 213

[11] WOO-CHEOL KWON AND TAEWHAN KIM: Optimal voltage allocation techniques for dynamically variable voltage processors. *ACM Trans. Embed. Comput. Syst.*, 4(1):211–230, February 2005. Preliminary version in DAC'03. [doi:10.1145/1053271.1053280] 210, 213

[12] MINMING LI, BECKY JIE LIU, AND FRANCES F. YAO: Min-energy voltage allocation for tree-structured tasks. *J. Combinatorial Optimization*, 11(3):305–319, 2006. [doi:10.1007/11533719_30] 210

[13] MINMING LI, ANDREW C. YAO, AND FRANCES F. YAO: Discrete and continuous min-energy schedules for variable voltage processors. In *Proc. Nat. Acad. Sci. USA*, volume 103, pp. 3983–3987, 2006. 213

[14] MINMING LI AND FRANCES F. YAO: An efficient algorithm for computing optimal discrete voltage schedules. *SIAM J. Comput.*, 35:658–671, 2005. [doi:10.1137/050629434] 210, 213

[15] KIRK PRUHS: Competitive online scheduling for server systems. *SIGMETRICS Perform. Eval. Rev.*, 34(4):52–58, 2007. [doi:10.1145/1243401.1243411] 214

[16] FRANCES YAO, ALAN DEMERS, AND SCOTT SHENKER: A scheduling model for reduced CPU energy. In *Proc. 36th FOCS*, pp. 374–382. IEEE Comp. Soc. Press, 1995. [doi:10.1109/SFCS.1995.492493] 210, 211, 213

[17] HAN-SAEM YUN AND JIHONG KIM: On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Trans. Embedded Computing Systems*, 2(3):393–430, 2003. [doi:10.1145/860176.860183] 210

AUTHORS

Nikhil Bansal
Eindhoven University of Technology
The Netherlands
bansal@tue.nl
http://www.win.tue.nl/~nikhil/

Ho-Leung Chan
Department of Computer Science
The University of Hong Kong, Hong Kong
hlchan@cs.hku.hk
http://i.cs.hku.hk/~hlchan/

Dmitriy Katz
Research Staff Member
IBM T.J. Watson, Yorktown, New York
dimdim@alum.mit.edu

Kirk Pruhs
Computer Science Dept.
University of Pittsburgh, USA
kirk@cs.pitt.edu
http://www.cs.pitt.edu/~kirk/

ABOUT THE AUTHORS

NIKHIL BANSAL graduated from CMU in 2003; his advisor was Avrim Blum. His research has focused on approximation and online algorithms for scheduling and other optimization problems. He worked at the IBM T. J. Watson Research Center from 2003 to 2011, where he also managed the Algorithms Group for some time. He recently moved to the Netherlands, and is still getting used to the idea that he can bike to anywhere without being run down by cars and trucks.

HO-LEUNG CHAN obtained his Ph. D. from The University of Hong Kong in 2007; his advisor was Tak-Wah Lam. His research interest is mainly in energy-efficient scheduling and other scheduling problems.

DMITRIY KATZ graduated from MIT in 2008. His advisors were Dimitris Bertsimas and David Gamarnik. His research interests lie in several areas of Computer Science and Applied Mathematics, including questions of decidability, combinatorial counting, queueing theory, and optimization.

KIRK PRUHS received his Ph. D. in 1989 from the from the University of Wisconsin - Madison. He is currently a professor of computer science at the University of Pittsburgh. His current research interests are in algorithmic problems related to resource management, scheduling, and sustainable computing. Within the algorithmic scheduling community, he is widely recognized as the leading expert on Mafia.