Operations Research Letters 40 (2012) 180–184

Contents lists available at SciVerse ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Speed scaling for stretch plus energy

Daniel Cole[a,*], Sungjin Im[b], Benjamin Moseley[b], Kirk Pruhs[a]

[a] *Computer Science Department, University of Pittsburgh, United States*

[b] *Department of Computer Science, University of Illinois, United States*

ARTICLE INFO

Article history:
Received 22 August 2011
Accepted 19 February 2012
Available online 27 February 2012

Keywords:
Speed scaling
Scheduling
Energy
Online algorithms

ABSTRACT

We consider speed scaling problems where the objective is to minimize a linear combination of arbitrary scheduling objective $\mathcal{S}$, and energy $\mathcal{E}$. A natural conjecture is that there is an $O(1)$-competitive algorithm for $\mathcal{S}$ on a fixed speed processor if and only if there is an $O(1)$-competitive algorithm for $\mathcal{S}+\mathcal{E}$ on a processor with an arbitrary power function. We give evidence to support this conjecture by providing an $O(1)$-competitive algorithm for the objective of integer stretch plus energy.

1. Introduction

All major chip makers now produce speed scalable chips and associated power management software. The algorithmic speed scaling literature studies the interplay between the system's speed scaling policy, which determines the speed of the processor at each time, and the system's job selection policy, which determines the task/job that is processed at each time. The processor is assumed to have a power function $P(s)$ specifying the power used by the processor when running at speed s. The system is assumed to have conflicting dual objectives: a scheduling quality of service objective, and an energy objective. To reconcile these conflicting objectives, one major line of algorithmic speed scaling research considers the objective of minimizing a linear combination $\alpha\mathcal{S}+\beta\mathcal{E}$ of a scheduling objective $\mathcal{S}$ and energy consumption $\mathcal{E}$ [1–12]. Here, α and β specify the relative importance of the scheduling objective and energy. Intuitively, the optimal schedule for the objective $\alpha\mathcal{S}+\beta\mathcal{E}$ is one that invests energy in such a way as to give the best resulting decrease in the scheduling objective $\mathcal{S}$ until the investment of an additional unit of energy cannot result in a schedule that reduces the scheduling objective by more than $\frac{\beta}{\alpha}$. By rescaling the units of time and energy one may assume without loss of generality that $\alpha=\beta=1$. The most common measure of goodness for a scheduler is the worst case relative error (the competitive ratio) of a resulting schedule relative to the optimal schedule. Naturally, one seeks schedulers where the competitiveness is relatively small.

One natural question for either clairvoyant or non-clairvoyant schedulers is as follows. "For what scheduling objectives $\mathcal{S}$ is there a speed scaling algorithm that is constant competitive for $\mathcal{S}+\mathcal{E}$ when the power function is arbitrary?". A constant competitive algorithm is achievable for some scheduling objectives. For example, for the scheduling objective of integer flow, there is a clairvoyant speed scaling algorithm that is constant competitive for an arbitrary power function [4]. This algorithm uses Shortest Remaining Processing Time (SRPT) for job selection and, at each time, chooses the processor speed that results in power consumption equal to the number of unfinished jobs. For the scheduling objective of fractional weighted flow, there is a clairvoyant speed scaling algorithm that has constant competitiveness for an arbitrary power function [4]. This algorithm uses Highest Density First (HDF) for job selection and, at each time, sets the processor speed so that the power consumption equals the total fractional weight of the unfinished jobs. Both of these speed scaling algorithms use the "natural" speed scaling algorithm that balances the rate of increase of the scheduling objective with the rate of increase of the energy objective by setting the instantaneous increase of the power consumption equal to the instantaneous increase of the scheduling objective $\mathcal{S}$. For some scheduling objectives, constant competitiveness is only achievable when restrictions are placed on the growth rate of the power function P.

In [6], Chan et al. showed that non-clairvoyant speed scaling algorithms cannot achieve constant competitiveness for the objective of integer flow plus energy if the power function is too steep, although constant competitiveness is achievable if P is assumed to be bounded by a polynomial with a constant degree. They extended the previous result in [13] that showed there is no

* Correspondence to: Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, United States.

E-mail address: dcc20@cs.pitt.edu (D. Cole).

0167-6377/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2012.02.003

non-clairvoyant algorithm with constant competitiveness for the objective of integer flow time on a fixed speed processor. Given the evidence to date, we make the following bold, yet natural, conjecture.

Conjecture. *There exists an online scheduling algorithm that is constant competitive on a fixed speed processor for the scheduling objective $\mathcal{S}$ if and only if there exists an online scheduling algorithm that is constant competitive for the scheduling objective $\mathcal{S} + \mathcal{E}$ when an arbitrary power function is considered.*

Unfortunately, it seems hard to imagine that the conjecture can be formally proven. A difficulty in showing the conjecture is that it does not seem possible to use the fact that constant competitiveness is achievable by an algorithm A for the scheduling objective $\mathcal{S}$ in the fixed speed setting as a black box to deduce anything about A in the speed scaling setting. Indeed, all previous analysis of algorithms in the speed scaling setting must know *why* A was constant competitive in the fixed speed setting. Thus, to gain confidence of the truth of this conjecture perhaps one must settle for showing that the conjecture holds for particular natural scheduling objectives.

One of the most obvious scheduling objectives to consider to further test this conjecture is the integer stretch objective. It is known that SRPT is 2-competitive for integer stretch in the fixed speed setting [14]. The integer stretch of a job is the job's integer flow time divided by the size of the job. This metric measures how much the completion of the job was slowed down relative to what it would have been on a dedicated unit speed processor. In cases where the user has some idea of the size of the job (for example when downloading static files from a web server, the user may know that video files are generally larger and will take longer than text files), integer stretch may be a better measure of the user perceived quality of service than integer flow.

In Section 3, we show that constant competitiveness is achievable by a clairvoyant speed scaling algorithm A for the objective of integer stretch plus energy. The job selection algorithm is a variation of HDF, and the speed scaling algorithm is a variation of the natural algorithm. This can be viewed as evidence supporting the conjecture. One difficulty in analyzing the competitiveness of this algorithm is that there is no optimal online scheduling algorithm for the objective of integer stretch in the setting of a fixed speed processor [14]. The analyses in [4] of speed scaling algorithms for arbitrary power functions critically uses the fact that the scheduling algorithms considered (SRPT for the scheduling objective of integer flow and HDF for the scheduling objective of fractional weighted flow) are not only optimal, but have the additional property that they do not fall any further behind an arbitrary algorithm when starting from arbitrary configurations. Since we cannot use the optimality of the scheduling algorithm in the fixed speed setting, our analysis is necessarily very different from the amortized local competitiveness analyses in [4]. By contrast, we introduce a new algorithm A and our analysis bounds the integer stretch of the algorithm A by its fractional stretch and the aggregate integer stretch accumulated from jobs during times they are processed by A. This allows us to focus on the scheduling objective of fractional weighted flow (a generalization of fractional stretch) plus energy. For this objective HDF is known to be 2-competitive. We bound the integer stretch of the currently running job by the optimal schedule's integer stretch for this job and the energy the optimal schedule uses while running this job.

1.1. Related results

Pruhs et al. [15] considered the problem of minimizing integer flow subject to an energy constraint, and gave an efficient offline algorithm for the case of unit size jobs. The problem of minimizing integer flow time plus energy was first proposed in [1]. They considered the unbounded speed scaling model with power function $P(s) = s^\alpha$, and gave an $O(1)$-competitive algorithm for the case of unit jobs. These results were improved upon in [5] which showed that the natural speed scaling algorithm is 4-competitive for integer flow time plus energy for unit jobs. Bansal et al. [5] further considered the objective of fractional weighted flow plus energy and gave an $O(\alpha/\log(\alpha))$-competitive algorithm which, using standard resource augmentation, implies an $O((\alpha/\log(\alpha))^2)$-competitive algorithm for integer weighted flow plus energy. Lam et al. [11] gave an $O(\alpha/\log(\alpha))$-competitive algorithm for integer flow plus energy for arbitrary sized jobs. In the case that there is an upper bound on the speed of the processor, Bansal et al. [3] gave an $O(\alpha/\log(\alpha))$-competitive algorithm for fractional weighted flow time plus energy for arbitrary sized jobs.

Speed scaling with arbitrary power functions was first considered in [4]. Bansal et al. [4] shows that the speed scaling algorithm that runs at a power equal to 1 plus the number of active jobs, and that uses SRPT for job selection is a 3-competitive algorithm for the objective of integer flow plus energy. Bansal et al. [4] also shows that the speed scaling algorithm that runs at power equal to the fractional weighted flow of all unfinished jobs, and that uses HDF for job selection is a 2-competitive algorithm for the objective of fractional weighted flow plus energy. Andrew et al. [2] shows that one could modify the analysis in [4] to show that the speed scaling algorithms that runs at a power equal to the number of unfinished jobs, and that uses SRPT for job selection is 2-competitive for the objective of integer flow plus energy. Andrew et al. [2] further showed that no "natural" online speed scaling algorithm can be better than 2-competitive. Andrew et al. [2] also showed that, for $P(s) = s^\alpha$ power functions, the natural speed scaling algorithm with processor sharing (PS) for job selection is $\max\{4\alpha - 2, 2(2 - 1/\alpha)^\alpha\}$-competitive. Gupta et al. [8,9] show how to extend the analyses in [4] to a setting of power heterogeneous multiprocessors.

2. Preliminaries

An instance I consists of n jobs that arrive over time. Job i has release (arrival) time r_i, work/size p_i, and possibly a weight w_i. The density of a job is its weight divided by total size, that is w_i/p_i. A clairvoyant online scheduler is not aware of job i until time r_i, at which time it learns p_i and w_i. A nonclairvoyant scheduler does not learn p_i at time r_i. At each time the scheduler specifies a job to process and a speed for the processor. Preemption is allowed, that is, a job may be suspended and later restarted from the point of suspension. A job i is completed once p_i units of work have been processed on i. The speed is the rate at which work is processed. If job i of work p_i is run at a constant speed s until completion then job i will complete p_i/s time units after being started.

For a fixed schedule σ, we let C_i^σ denote job i's completion time. The flow (time) $F_i^\sigma = C_i^\sigma - r_i$ of job i is the time that elapses after the job arrives until being completed. When the considered schedule σ is clear in the context, we may omit the superscript σ. Job i's integer stretch is F_i/p_i, and its integer weighted flow is w_iF_i. The integer stretch of a job can be viewed as a special case of integer weighted flow where the weight is the reciprocal of the job's size. The integer flow of a schedule is $\sum_{i=1}^n F_i$. The integer stretch of a schedule is $\sum_{i=1}^n F_i/p_i$, and the integer weighted flow is $\sum_{i=1}^n w_iF_i$. If the unfinished work of job i at time t is $p_i(t)$ then the fractional size of job i at time t is $p_i(t)/p_i$ and the fractional weight of job i at time t is $w_i(p_i(t)/p_i)$. The fractional flow of job i is defined to be $\int_{r_i}^\infty p_i(t)/p_i \, \mathrm{d}t$ and the fractional weighted flow of job i is $\int_{r_i}^\infty w_i(p_i(t)/p_i) \, \mathrm{d}t$. The fractional flow time and fractional weighted flow time of a schedule are the sum over all jobs of the fractional flow time or fractional weighted flow time respectively. The fractional stretch of a schedule is the same as the fractional weighted flow time where the weight of job i is set to $1/p_i$.

We adopt the speed scaling model originally proposed in [4] that essentially allows the power function to be arbitrary. In particular the power function may have a maximum power consumption rate, and hence maximum speed. The only real restriction on the power function is that it is piecewise continuous, differentiable, and integrable. It is shown in [4] that without loss of generality, we can assume that the power function $P: [0, s_{\max}] \to [0, P_{\max}]$ is continuous, differentiable, and convex, such that $P(0) = 0$ and $P(s_{\max}) = P_{\max}$. Here $s_{\max}$ ($P_{\max}$) denote the maximum possible speed (power). By definition $P_{\max} = P(s_{\max})$. The energy (consumption) of a schedule is the power usage integrated over time.

We use $\widetilde{\mathrm{WF}}$, S, $\widetilde{S}$, and E to denote the objectives of fractional weighted flow, integer stretch, fractional stretch, and energy, respectively. We use $\widetilde{\mathrm{WF}} \oplus E$ and $S \oplus E$ to denote the objective of fractional weighted flow plus energy and integer stretch plus energy, respectively. For an algorithm A we denote the schedule output by A on input I by $A(I)$. We use $OPT(I)$ to denote the optimal schedule for the objective under consideration. For example, $\widetilde{\mathrm{WF}} \oplus E\,\langle A(I)\rangle$ denotes the fractional weighted flow plus energy for the schedule output by algorithm A on input I, and $S \oplus E\,\langle OPT(I)\rangle$ denotes the optimal integer stretch plus energy for input I.

3. An algorithm for stretch plus energy with bounded competitiveness

In this section, we give a clairvoyant speed scaling algorithm A, and show that A is $O(1)$ competitive for the objective of integer stretch plus energy on a processor with an arbitrary power function. We start by giving a definition of the online algorithm from [4], which we call BCP. BCP takes as input an online sequence of jobs with release times, sizes, and weights.

Definition of online algorithm BCP. At any time t, BCP always runs the unfinished job with the highest density at power

$$P_b(t) = \min\{w_b(t), P_{\max}\}$$

where $P_{\max}$ is the maximum power and $w_b(t)$ is the sum of the fractional weights of all unfinished jobs for BCP at time t.

We will assume that in case of equal density jobs, BCP breaks ties in favor of earlier released jobs. One of the two main results in [4] is Theorem 1, that BCP is 2-competitive for the objective of fractional weighted flow plus energy.

Theorem 1 *([4]). For all inputs I, $\widetilde{WF} \oplus E\,\langle BCP(I)\rangle \le 2 \cdot \widetilde{WF} \oplus E\,\langle OPT(I)\rangle$.*

At a high level, A is the same as BCP with two important differences. The first difference is that A rounds the weight of each job up so that the density is an integer power of some constant $f > 1$, so that $\frac{1}{p_j} \le w_j < \frac{f}{p_j}$. This is to prevent the algorithm A from preempting between jobs of similar densities. The second difference is that A never lets the power fall below one over the work of the active job. Let I be an arbitrary instance of jobs with release times and sizes. Let I' be the corresponding instance where additionally all jobs j have weights w_j, where w_j is the minimal real number, not less than $1/p_j$, such that w_j/p_j is an integer power of f. The algorithm A can either be thought of as running on an unweighted instance, where the algorithm instantiates the weights, or on a weighted instance in which weights are provided as part of the input.

Definition of online algorithm A. When considering an unweighted instant I, A assigns jobs the weights that they would have in instance I'. At any time t, A runs the highest density job, breaking ties in favor of earlier released jobs, at power

$$P_a(t) = \min\left\{\max\left\{w_a(t), \frac{1}{p_{a(t)}}\right\}, P_{\max}\right\}$$

where $a(t)$ is the job being processed at time t, $p_{a(t)}$ is the size of $a(t)$, and $w_a(t)$ is the sum of the fractional weights of all unfinished jobs at time t.

We show that, by picking $f \approx 2.015$, A is approximately 9.414-competitive for the objective of integer stretch plus energy. Our analysis can be summarized as the following sequence of bounding steps:

$$\begin{aligned}
&S \oplus E\langle A(I)\rangle \\
&\quad\le \widetilde{\mathrm{WF}} \oplus E\langle A(I')\rangle + \left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right)\int_t \frac{1}{p_{a(t)}} \quad \text{(Lemma 2)}\\
&\quad\le \widetilde{\mathrm{WF}} \oplus E\langle A(I')\rangle \\
&\qquad + \left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right) S \oplus E\langle OPT(I)\rangle \quad \text{(Lemma 3)}\\
&\quad\le (2f+1)\cdot S \oplus E\langle OPT(I)\rangle \\
&\qquad + \left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right) S \oplus E\langle OPT(I)\rangle \quad \text{(Lemma 4)}\\
&\quad= \left(2f+2+\frac{\sqrt{f}}{\sqrt{f}-1}\right) S \oplus E\langle OPT(I)\rangle \\
&\quad\approx 9.414 \cdot S \oplus E\langle OPT(I)\rangle.
\end{aligned}$$

Before proving each of the lemmas used above, we give some intuitive explanation of the lemmas. Lemma 2 shows that the integer stretch of A is bounded by the fractional weighted flow of A plus the aggregate integer stretch accumulated from jobs while they are run by A. The proof relies on the fact that densities of the preempted jobs in A's schedule form a geometric sequence. Lemma 3 shows via a charging scheme that the aggregate integer stretch accumulated from jobs while they are run by A is at most the optimal integer stretch plus energy. Intuitively, this lemma can be explained as follows. For each job j, consider the times that j is processed by A and OPT. If OPT processes j faster than A by using more power, j's integer stretch can be charged to OPT's power usage. Otherwise, job j, as an active job, contributes to A's integer stretch less than OPT's integer stretch. Lemma 4 relates the fractional weighted flow plus energy for A to the optimal integer stretch plus energy. Recall that A uses the same speed scheduling policy as BCP except that it runs at power at least $1/p_{a(t)}$ at any time t. Let us call the time periods that A uses power exactly $1/p_{a(t)}$ as minimum power periods and the other time periods as normal time periods. By mimicking the analysis of BCP, we can bound the fractional flow for A, plus the energy used by A during normal time periods. We separately bound the energy used by A during minimum power periods by the integer stretch of the active jobs. Before preceding to these lemmas, in Lemma 1 we make an intuitive observation that the optimal fractional weighted flow for instance I' is at most f times the optimal integer stretch plus energy for instance I.

Lemma 1. $\widetilde{WF} \oplus E\left\langle OPT(I')\right\rangle \le f \cdot S \oplus E\,\langle OPT(I)\rangle$.

Proof. First note that $\widetilde{\mathrm{WF}} \oplus E\left\langle OPT(I')\right\rangle \le f \cdot \widetilde{S} \oplus E\,\langle OPT(I)\rangle$ since each weight w_j in I' is at most f/p_j. Further it follows that $\widetilde{S} \oplus E\,\langle OPT(I)\rangle \le S \oplus E\,\langle OPT(I)\rangle$ since the fractional stretch of any schedule is no more than the integer stretch of that schedule. □

Lemma 2. $S \oplus E\langle A(I)\rangle \le \widetilde{WF} \oplus E\langle A(I')\rangle + \left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right)\int_t \frac{1}{p_{a(t)}}$.

Proof. Throughout this lemma, we will implicitly use the fact that A creates the same schedule for the instance I and the instance I' by definition of A. To show the lemma, we focus on showing the stronger statement that $S\langle A(I)\rangle \leq \widetilde{\mathrm{WF}}\langle A(I')\rangle + \left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right)\int_t \frac{1}{p_{a(t)}}$. Notice that this is strictly a stronger statement because the energy used by A is the same for I' and I. Fix a time t. We partition the unfinished jobs in A's schedule at time t into three sets: the running job, $a(t)$, the set $A_u(t)$ of jobs that have not been processed by A, and the set $A_p(t)$ of jobs that have been partially processed by A, excluding the running job $a(t)$. It is sufficient to establish the following invariant:

$$\sum_{i\in A_u(t)}\frac{1}{p_i}+\sum_{i\in A_p(t)}\frac{1}{p_i}+\frac{1}{p_{a(t)}} \leq \sum_{i\in A_u(t)} w_i\frac{p_i(t)}{p_i}+\left(1+\frac{\sqrt{f}}{\sqrt{f}-1}\right)\frac{1}{p_{a(t)}}.$$

Proving this invariant is sufficient to prove the lemma because the left hand side of the inequality is the instantaneous increase in the stretch objective for A's schedule at time t and $\sum_{i\in A_u(t)} w_i\frac{p_i(t)}{p_i}$ is the instantaneous increase in the fractional weighted flow time of A's schedule at time t. Thus, if this inequality can be shown for all times t, then by integrating over time the lemma follows.

By definition $p_i(t)/p_i = 1$ for unprocessed jobs. Also, $w_i \geq 1/p_i$ by the definition of I'. Knowing this, we have

$$\sum_{i\in A_u(t)}\frac{1}{p_i} \leq \sum_{i\in A_u(t)} w_i\frac{p_i(t)}{p_i}.$$

Thus to establish our invariant, it is sufficient to show that

$$\sum_{i\in A_p(t)}\frac{1}{p_i} \leq \frac{\sqrt{f}}{\sqrt{f}-1}\frac{1}{p_{a(t)}}.$$

Consider the jobs $d(0),\ldots,d(m)$ in $A_p(t)$ by increasing order of arrival time where $m=|A_p(t)|$. By the definition of I', any two jobs either have equal densities or their densities differ by at least a factor of f. Knowing that all jobs in $A_p(t)$ have been partially processed by A and that A breaks ties in favor of jobs with earlier release dates, it must be the case that $f\frac{w_{d(i)}}{p_{d(i)}} \leq \frac{w_{d(i+1)}}{p_{d(i+1)}}$ for $0\leq i\leq m-1$ and $f\frac{w_{d(m)}}{p_{d(m)}} \leq \frac{w_{a(t)}}{p_{a(t)}}$. Hence we have $\frac{w_{d(i)}}{p_{d(i)}} \leq \frac{1}{f^{m-i+1}}\frac{w_{a(t)}}{p_{a(t)}}$. For any job j, by definition of w_j, $\frac{1}{p_j}\leq w_j < \frac{f}{p_j}$. Knowing this, we have that $\frac{1}{(p_{d(i)})^2} \leq \frac{w_{d(i)}}{p_{d(i)}} \leq \frac{1}{f^{m-i+1}}\frac{w_{a(t)}}{p_{a(t)}} \leq \frac{1}{f^{m-i}}\frac{1}{(p_{a(t)})^2}$. Thus we have,

$$\sum_{i\in A_p(t)}\frac{1}{p_i} = \sum_{i=0}^{m}\frac{1}{p_{d(i)}} \leq \sum_{i=0}^{m}\frac{1}{\left(\sqrt{f}\right)^{m-i}}\frac{1}{p_{a(t)}} \leq \frac{1}{p_{a(t)}}\sum_{i=0}^{\infty}\frac{1}{\left(\sqrt{f}\right)^{i}} = \frac{\sqrt{f}}{\sqrt{f}-1}\frac{1}{p_{a(t)}}. \quad \square$$

Lemma 3. $\int_t \frac{1}{p_{a(t)}} \leq S\oplus E\,\langle OPT(I)\rangle$.

Proof. Consider any arbitrary job j with total work p_j, and an arbitrary infinitesimal portion of that work dp_j. Let dt_A be the amount of time that A spends actively working on p_j to complete the dp_j portion of p_j's work, and let dt_O be the amount of time that OPT spends actively working on p_j to complete the dp_j portion of p_j's work. We call dt_A/p_j the contribution of dp_j to $\int_t \frac{1}{p_{a(t)}}$. Our goal is to bound the contribution of dp_j by the optimal solution's cost when the optimal solution processes the dp_j portion of p_j's work. Once this is shown, by integrating over all portion's of p_j's work and summing over all jobs, the lemma follows. We consider two cases depending on the relationship of dt_A and dt_O.

First consider the case where $dt_A \leq dt_O$. In this case, we can charge the contribution of dp_j, to the integer stretch penalty OPT incurs while processing the dp_j portion of p_j. Indeed, the integer stretch penalty OPT incurs for job j is dt_O/p_j during this time. Now consider the other case where $dt_A > dt_O$. In this case, the convexity of the power function implies that A uses no more energy than OPT while processing the dp_j portion of p_j. In particular, A's energy usage is strictly smaller $P_{\max}$. Thus, by definition of A, A runs the processor using power at least $1/p_j$ the entire time the dp_j portion of p_j is being processed. This implies that the energy used by A to complete the work dp_j is at least the integer stretch penalty dp_j/p_j incurred by A. Hence we can charge dp_j/p_j to the energy that OPT uses to process dp_j. $\square$

Lemma 4. $\widetilde{WF}\,\oplus E\langle A(I')\rangle \leq (2f+1)\cdot S\oplus E\langle OPT(I)\rangle$.

Proof. For instances where densities are integer powers of f, the only difference between the algorithm A and the algorithm BCP is that on some configurations A would run faster than BCP. In particular, at times when $w_a(t) < 1/p_{a(t)} \leq P_{\max}$, A will run at power $1/p_{a(t)}$ while BCP would only run at power $w_a(t)$. Recall that these times are called the minimum power periods for A.

The analysis of BCP in [4] uses an amortized local competitiveness argument. That is, it gives a potential function $\Phi(t)$ so that the following invariant holds at all times t:

$$w_a(t)+P_a(t)+\frac{d\Phi(t)}{dt} \leq 2(w_o(t)+P_o(t))$$

where $P(t)$ is the power used by A, $w_o(t)$ is the unfinished fractional weight for the optimal schedule and $P_o(t)$ is the power used in the optimal schedule. This invariant establishes that for BCP, $\widetilde{\mathrm{WF}}\oplus E\langle \mathrm{BCP}(I')\rangle \leq 2\cdot \mathrm{WF}\oplus E\langle OPT(I')\rangle$. Hence by Lemma 1, $\widetilde{\mathrm{WF}}\oplus E\langle \mathrm{BCP}(I')\rangle \leq 2f\cdot S\oplus E\,\langle OPT(I)\rangle$. If we attempted to repeat this analysis with the algorithm A, the only problem is that during the minimum power periods, A might run faster than BCP, meaning that this analysis would not account for all the energy used by A during the minimum power periods. Therefore, the only task left is to bound the total power used by A during the minimum power periods. Note that any time t that is a minimum power period, A uses power $1/p_{a(t)}$. Therefore, the total energy consumption by A during all the minimum power periods is bounded by $\int_t 1/p_{a(t)}$. By Lemma 3, it is at most $S\oplus E\,\langle OPT(I)\rangle$. Combining this quantity with the upper bound obtained following the BCP analysis completes the proof. $\square$

Acknowledgments

This was supported in part by NSF grants CCF-0830558 and 1115575, and an IBM Faculty Award. It was partially supported by NSF grants CCF-0728782 and CCF-1016684.

References

[1] S. Albers, H. Fujiwara, Energy-efficient algorithms for flow time minimization, ACM Transactions on Algorithms 3 (4) (2007) 49:1–49:17.
[2] L.L. Andrew, M. Lin, A. Wierman, Optimality, fairness, and robustness in speed scaling designs, in: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2010, pp. 37–48.
[3] N. Bansal, H.-L. Chan, T.-W. Lam, L.-K. Lee, Scheduling for speed bounded processors, International Colloquium on Automata, Languages and Programming, Part I (2008) 409–420.
[4] N. Bansal, H.-L. Chan, K. Pruhs, Speed scaling with an arbitrary power function, in: ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 693–701.
[5] N. Bansal, K. Pruhs, C. Stein, Speed scaling for weighted flow time, SIAM Journal on Computing 39 (4) (2009) 1294–1308.
[6] H.-L. Chan, J. Edmonds, T.-W. Lam, L.-K. Lee, A. Marchetti-Spaccamela, K. Pruhs, Nonclairvoyant speed scaling for flow and energy, Algorithmica 61 (3) (2011) 507–517.
[7] H.-L. Chan, J. Edmonds, K. Pruhs, Speed scaling of processes with arbitrary speedup curves on a multiprocessor, Theory of Computing Systems 49 (4) (2011) 817–833.
[8] A. Gupta, R. Krishnaswamy, K. Pruhs, Nonclairvoyantly scheduling power-heterogeneous processors, in: International Conference on Green Computing, 2010, pp. 165–173.

[9] A. Gupta, R. Krishnaswamy, K. Pruhs, Scalably scheduling power-heterogeneous processors, International Colloquium on Automata, Languages and Programming, Part I (2010) 312–323.
[10] T.-W. Lam, L.-K. Lee, H.-F. Ting, I.K. To, P.W. Wong, Sleep with guilt and work faster to minimize flow plus energy, International Colloquium on Automata, Languages and Programming, Part I (2009) 665–676.
[11] T.-W. Lam, L.-K. Lee, I.K. To, P.W. Wong, Speed scaling functions for flow time scheduling based on active job count, in: European Symposium on Algorithms, 2008, pp. 647–659.
[12] T.-W. Lam, L.-K. Lee, I.K.K. To, P.W.H. Wong, Nonmigratory multiprocessor scheduling for response time and energy, IEEE Transactions on Parallel and Distributed Systems 19 (11) (2008) 1527–1539.
[13] R. Motwani, S. Phillips, E. Torng, Non-clairvoyant scheduling, Theoretical Computer Science 130 (1) (1994) 17–47.
[14] S. Muthukrishnan, R. Rajaraman, A. Shaheen, J.E. Gehrke, Online scheduling to minimize average stretch, SIAM Journal on Computing 34 (2) (2005) 433–452.
[15] K. Pruhs, P. Uthaisombut, G. Woeginger, Getting the best response for your erg, ACM Transactions on Algorithms 4 (3) (2008) 38:1–38:17.