Non-clairvoyant Speed Scaling for Weighted Flow Time

Sze-Hang Chan[1], Tak-Wah Lam[1], and Lap-Kei Lee[2]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
[2] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Abstract. We study online job scheduling on a processor that can vary its speed dynamically to manage its power. We attempt to extend the recent success in analyzing total unweighted flow time plus energy to total weighted flow time plus energy. We first consider the non-clairvoyant setting where the size of a job is only known when the job finishes. We show an online algorithm WLAPS that is $8\alpha^2$-competitive for weighted flow time plus energy under the traditional power model, which assumes the power $P(s)$ to run the processor at speed s to be s^α for some $\alpha > 1$. More interestingly, for any arbitrary power function $P(s)$, WLAPS remains competitive when given a more energy-efficient processor; precisely, WLAPS is $16(1 + \frac{1}{\epsilon})^2$-competitive when using a processor that, given the power $P(s)$, can run at speed $(1+\epsilon)s$ for some $\epsilon > 0$. Without such speedup, no non-clairvoyant algorithm can be $O(1)$-competitive for an arbitrary power function [8]. For the clairvoyant setting (where the size of a job is known at release time), previous results on minimizing weighted flow time plus energy rely on scaling the speed continuously over time [5–7]. The analysis of WLAPS has inspired us to devise a clairvoyant algorithm LLB which can transform any continuous speed scaling algorithm to one that scales the speed at discrete times only. Under an arbitrary power function, LLB can give an $4(1 + \frac{1}{\epsilon})$-competitive algorithm using a processor with $(1 + \epsilon)$-speedup.

1 Introduction

To reduce energy usage, manufacturers like Intel and IBM are now producing processors that can support *dynamic speed scaling*, which would allow operating systems to manage the power by scaling the processor speed dynamically. The theoretical study of speed scaling was initiated by Yao, Demers and Shenker [18]. They considered a model where a processor can vary the speed s dynamically, and it consumes energy at the rate s^α, where $\alpha > 1$ is a constant (commonly believed to be 2 or 3 [3, 16]). Running jobs slower is more energy-efficient, yet it takes longer time. Taking speed scaling and energy usage into consideration makes job scheduling more complicated than before. A scheduling algorithm needs two components: a job selection policy to determine which job to run, and a speed scaling policy to determine the speed to run the job. The challenge arises from the conflicting objectives of optimizing some quality of service (QoS) of the schedule and minimizing the energy usage.

M. de Berg and U. Meyer (Eds.): ESA 2010, Part I, LNCS 6346, pp. 23–35, 2010.

Flow and energy. The past few years have witnessed several interesting results on online scheduling for optimizing the tradeoff between energy usage and flow time (e.g., [1, 5–8, 12–14]). The flow time (or simply the flow) of a job is the time elapsed since the job is released until it is completed. Assuming jobs are equally important, it is natural to find a schedule that minimizes the total flow time (also known as minimizing the total/average response time). In general, jobs have varying importance or weights, and it is more meaningful to minimize the total weighted flow time. In the speed scaling model, minimizing total flow time and energy usage are orthogonal objectives. To understand their tradeoff, Albers and Fujiwara [1] initiated the study of minimizing a linear combination of flow and energy. The intuition is that, from an economic viewpoint, users are willing to pay a certain (say, ρ) units of energy to reduce one unit of flow time. By changing the units of time and energy, one can further assume $\rho = 1$ and thus wants to minimize flow plus energy, or in general, weighted flow plus energy.

Clairvoyant scheduling for flow plus energy. Most of the previous work on minimizing flow plus energy focused on the online setting where the size of a job is known at its release time. This is known as the clairvoyant setting. Bansal, Pruhs and Stein [7] were the first to consider jobs with arbitrary weights, and they give an online algorithm BPS that is $O((\frac{\alpha}{\ln \alpha})^2)$-competitive for minimizing weighted flow plus energy. Very recently, Bansal, Chan and Pruhs [6] improved the analysis of BPS, which implies that BPS is $O(\frac{\alpha}{\ln \alpha})$-competitive.

The BPS algorithm scales the speed as a function of the fraction of unfinished work and thus it keeps changing the speed continuously over time. Practically speaking, it is more desirable to change the speed only at discrete times (say, at job arrival or completion). Focusing on jobs with unit-weight, Lam et al. [14] showed a competitive discrete-time-scaling algorithm called AJC (Active Job Count), which scales the speed as a function of the number of active jobs. When coupled with the job selection policy SRPT (shortest remaining processing time), it is $O(\frac{\alpha}{\log \alpha})$-competitive for (unweighted) flow plus energy. Recently, Bansal, Chan and Pruhs [6] gave a tighter analysis of AJC, showing that the competitive ratio is at most 3 (when $\alpha = 3$, the $O(\frac{\alpha}{\log \alpha})$ bound in [14] is equal to 3.25). More importantly, they extend the analysis to a more general model where the speed-to-power function can be an arbitrary function. This makes the results of speed scaling more robust than before. Andrew et al. [2] have further improved the ratio to 2. Extending the work of [2, 6, 14] to jobs with arbitrary weights is non-trivial. It is natural to generalize AJC to AJW (active job weight), which scales the speed according to the total weight of active jobs. However, even assuming the power function in the form s^α, it has been open whether AJW (plus a job selection policy like HDF) or any discrete-time-scaling algorithm can be competitive for minimizing weighted flow plus energy, let alone for an arbitrary power function.

Non-clairvoyant scheduling for flow plus energy. All of the above results assume clairvoyance. In the non-clairvoyant setting, the size of a job is only known when the job is completed. This is a natural assumption from the viewpoint of operating systems. Non-clairvoyant flow time scheduling (on

a fixed-speed processor) has been an interesting problem itself (e.g., [11, 15]). Chan et al. [8] were the first to study non-clairvoyant speed scaling. They focused on unit-weight jobs and considered an algorithm LAPS (Latest Arrival Processor Sharing) which scales the speed as AJC and selects some most recently released jobs to share the processor. LAPS is $O(\alpha^3)$-competitive for (unweighted) flow plus energy. Furthermore, it is shown that no algorithm can be $O(1)$-competitive for an arbitrary power function. Recently, Chan et al. [9] improved the competitive ratio of LAPS to $O(\frac{\alpha^2}{\log \alpha})$. For weighted flow plus energy, Chan et al. [10] showed that a round robin policy plus AJW is $O(3^\alpha)$-competitive. Note that all these results assume that the power function is in the form s^α.

Our contribution. The main result of this paper is about a new non-clairvoyant algorithm called WLAPS for minimizing weighted flow plus energy. WLAPS attempts to generalize LAPS [8, 9] to jobs with arbitrary weights. WLAPS uses the speed scaling policy AJW, i.e., the speed is a function of the total weight of active jobs. Like LAPS, WLAPS gives priority to some most recently released jobs. But WLAPS does not schedule a constant fraction of active jobs; instead it ensures that the jobs selected have a total weight equal to a constant fraction of the total weight of all active jobs, and they share the processor according to their job weights. Furthermore, as job weights are arbitrary, they may not make up exactly the required fraction. Thus, we sometimes need to modify the weight of some job to avoid under- or over-scheduling.

Using a slightly complicated potential analysis, we show that for a power function in the form s^α, WLAPS is $8\alpha^2$-competitive for weighted flow plus energy. We also analyze WLAPS with an arbitrary power function $P(s)$. In view of the lower bound result in [8], we consider giving WLAPS a more energy-efficient processor which, when given the power $P(s)$, can run at speed $(1+\epsilon)s$ for some $\epsilon > 0$. We call such a processor a $(1+\epsilon)$-speedup processor.[1] We prove that WLAPS is $16(1+\frac{1}{\epsilon})^2$-competitive when given a $(1+\epsilon)$-speedup processor. The main difficulty here is how to lower bound the optimal offline algorithm.

Table 1. Results on non-clairvoyant speed scaling for minimizing flow plus energy

	Unit-weight jobs	Arbitrary-weight jobs
Traditional power function (s^α)	$O(\frac{\alpha^2}{\log \alpha})$-competitive [9]	$8\alpha^2$-competitive **(new)**
Arbitrary power function	unbounded competitive ratio [8] (if no speedup)	$16(1+\frac{1}{\epsilon})^2$-competitive **(new)** with $(1+\epsilon)$-speedup

Implication to clairvoyant speed scaling. Since the clairvoyant setting is a special case of the non-clairvoyant setting, WLAPS also gives us the first clairvoyant discrete-time-scaling result on weighted flow plus energy, and it is

[1] Note that speed-up processors here are not related to processes (or jobs) with arbitrary speed-up curves in [9]; the former refers to more energy-efficient processors, and the latter is job characteristics, specifying the degrees of parallelizability (i.e., the rate of processing a job as a function of the number of processors assigned to it).

valid for an arbitrary power function. In fact, we can further improve the ratios. In the analysis of WLAPS, when we attempt to lower bound the performance of the optimal offline algorithm, we have devised a novel clairvoyant algorithm LLB (latest lag behind) which can transform any (online/offline) algorithm into one that scales the speed according to AJW, and the cost (weighted flow plus energy) at most doubles. Therefore, we can effectively transform the BPS algorithm [6, 7] to a discrete-time-scaling algorithm using the AJW policy that is $O(\frac{\alpha}{\ln \alpha})$-competitive for a power function in the form s^α, and $4(1+\frac{1}{\epsilon})$-competitive for an arbitrary power function using a processor with $(1+\epsilon)$-speedup for any $\epsilon > 0$.

Bounded maximum speed. All the above-mentioned results are based on an assumption in [18] that the processor does not have a limit on the speed. Such an unbounded speed model is a convenient model to work with. Among others, it allows an online algorithm to catch up arbitrarily fast and recover from any over-conservative decision on speed. However, this is not a practical model. Recently, there has been a growing interest in algorithms that scale the speed between 0 and a given maximum speed T. Though not straightforward, most unbounded-speed algorithms have been shown to work in the bounded speed model when the power function is in the form s^α.

For unit-weight jobs, the clairvoyant results on AJC by Lam et al. [14], Bansal et al. [6], and Andrew et al. [2] all remain valid when the maximum speed T is bounded. For the clairvoyant-weighted setting and the non-clairvoyant setting, the lower bound results on (fixed-speed) flow time scheduling [4, 15] imply that no online algorithm can be constant competitive (in terms of α) for weighted flow plus energy when T is bounded (if there is no resource augmentation). On the other hand, it has been shown that the algorithms BPS and LAPS when capped at maximum speed $(1+\delta)T$ for some $\delta > 0$, remains competitive as in the unbounded speed model [5, 10]. Note that relaxing the maximum speed is a less demanding form of resource augmentation than using a more efficient processor allowing speedup, and it is only applicable in the bounded speed model.

In this paper we also extend the results of WLAPS for weighted flow plus energy to the bounded speed model. For the traditional power function, WLAPS is $\max(8\alpha^2, 8(1+\frac{1}{\delta})^2)$-competitive when using a processor with maximum speed $(1+\delta)T$ for any $\delta > 0$. For an arbitrary power function, WLAPS remains $16(1+\frac{1}{\epsilon})^2$-competitive when using a processor with $(1+\epsilon)$-speedup for any $\epsilon > 0$. Note that if we only allow relaxing the maximum speed, no $O(1)$-competitive algorithm exists due to the lower bound result in the unbounded speed model [8]. Like [2, 6], our analysis exploits potential function. Yet, our analysis involves a technique of taking the maximum speed T into the design of potential functions.

2 Definitions and Notations

We study job scheduling on a single processor. Jobs with varying sizes arrive over time online; we have no information about a job before it arrives. For any job j, we use $r(j)$, $w(j)$ and $p(j)$ to denote its release time, weight and size

(work requirement), respectively. In the clairvoyant model, $p(j)$ is known at time $r(j)$. In the non-clairvoyant model, $p(j)$ is only known when the job is completed. The processor can vary its speed between 0 and a maximum speed T (where T is some fixed constant or ∞). When running at speed s, the processor processes s units of work per unit time and consumes energy at the rate $P(s)$. Preemption is allowed; a job can be preempted and later resumed at the point of preemption without any penalty. We consider the following two models of power function.

Traditional power model. In this model, we assume that the power function is $P(s) = s^\alpha$ for some $\alpha > 1$. Furthermore, we distinguish the two settings: *bounded speed* means that the speed s cannot exceed a fixed maximum speed T; and *unbounded speed* otherwise (i.e., $T = \infty$).

Arbitrary power model. In this model, we consider a power function P such that $P(0) = 0$, and P is defined, strictly increasing, strictly convex, continuous and differentiable at all speeds in $[0, T]$; if there is no maximum speed T, the speed range is $[0, \infty)$ and for any speed x, there exists x' such that $P(x)/x < P(s)/s$ for all $s > x'$ (otherwise the optimal speed scaling policy is to always run at the infinite speed and an optimal schedule is not well-defined). As shown in [6], it is possible to use any arbitrary power function to emulate such a power function P with an arbitrarily small increase in the competitive ratio. Thus, we focus on a power function P satisfying the above assumptions. We use Q to denote P^{-1}. Note that Q is strictly increasing and concave. E.g., if $P(s) = s^\alpha$, then $Q(x) = x^{1/\alpha}$.

Flow and energy. Consider any job set I and some schedule S of I. At any time t, for any job j, we let $q(j,t)$ be the remaining work of j at t. A job j is an *active job* if it has been released but not yet completed, i.e., $r(j) \le t$ and $q(j,t) > 0$. The *flow* of a job j is the time elapsed since j arrives and until it is completed, and the *weighted flow* $F(j)$ of j is $w(j)$ times its flow. The *total weighted flow* is $F = \sum_{j \in I} F(j)$, which is equivalent to $F = \int_0^\infty w(t)\,\mathrm{d}t$, where $w(t)$ is the total weight of active jobs at time t. The energy usage is $E = \int_0^\infty P(s(t))\,\mathrm{d}t$, where $s(t)$ is the processor speed at time t. The objective is to minimize the total weighted flow plus energy usage, denoted by $G = F + E$.

Overview of analysis. Throughout the paper, we often need to compare an algorithm ALG with another algorithm O (e.g., the optimal offline algorithm OPT). We will exploit amortization and potential functions. Let $G_\mathrm{a}(t)$ and $G_\mathrm{o}(t)$ denote the weighted flow plus energy incurred up to time t by ALG and O, respectively. We will drop the parameter t when it is clear that t is the current time. To show that ALG incurs at most c times the weighted flow plus energy of O, it suffices to define a potential function $\Phi(t)$ such that the following conditions hold: (i) *Boundary condition:* $\Phi = 0$ before any job is released and after all jobs are completed; (ii) *Discrete-event condition:* Φ is a continuous function except at some discrete times (e.g., when a job arrives, or when a job is completed by ALG or O), and Φ does not increase at such times; (iii) *Running condition:* at any other time, $\frac{\mathrm{d}G_\mathrm{a}(t)}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le c \cdot \frac{\mathrm{d}G_\mathrm{o}(t)}{\mathrm{d}t}$, where γ is a positive constant. The correctness of the analysis follows from integrating these conditions over time.

3 Non-clairvoyant Speed Scaling

This section considers a non-clairvoyant algorithm WLAPS for minimizing weighted flow plus energy. In Section 3.1, we show that under the traditional power model, WLAPS is $8\alpha^2$-competitive if the maximum speed T is unbounded; if T is bounded, WLAPS is $\max(8\alpha^2, 8(1+\frac{1}{\epsilon})^2)$-competitive when using a processor with maximum speed relaxed to $(1+\epsilon)T$ for any $\epsilon > 0$. In Section 3.2, we consider the arbitrary power model. Note that even in the unbounded speed setting, no $O(1)$-competitive algorithm exists [8]. Nevertheless, we show that WLAPS is $16(1+\frac{1}{\epsilon})^2$-competitive when using a $(1+\epsilon)$-speedup processor for any $\epsilon > 0$. This result holds no matter whether T is unbounded or bounded.

We now define the algorithm WLAPS. At any time t, we use $n_{\mathrm{a}}(t)$ and $w_{\mathrm{a}}(t)$ to denote respectively the number and total weight of active jobs. The active jobs at time t are referred to as $j_1, j_2, \ldots, j_{n_{\mathrm{a}}(t)}$, which are arranged in ascending order of their release times (ties are broken by job ids).

> **WLAPS.** Let $0 < \beta \leq 1$ be any real. Consider any time t. Let τ be the biggest integer such that the total weight of jobs $j_\tau, j_{\tau+1}, \ldots, j_{n_{\mathrm{a}}(t)}$ is at least $\beta w_{\mathrm{a}}(t)$. Define $w'(j_\tau)$, the adjusted weight of j_τ, to be $\beta w_{\mathrm{a}}(t) - [w(j_{\tau+1}) + \cdots + w(j_{n_{\mathrm{a}}(t)})]$. WLAPS processes $j_\tau, j_{\tau+1}, \ldots, j_{n_{\mathrm{a}}(t)}$ by splitting the processor speed in proportional to the adjusted weight of j_τ and the (original) weights of $j_{\tau+1}, \ldots, j_{n_{\mathrm{a}}(t)}$.
>
> WLAPS basically follows the speed scaling policy AJW, which sets the speed to $Q(w_{\mathrm{a}}(t))$. E.g., if $P(s) = s^\alpha$, then $Q(w_{\mathrm{a}}(t)) = w_{\mathrm{a}}(t)^{1/\alpha}$. In Sections 3.1 and 3.2, we will further refine the policy due to the presence of maximum speed and/or the need of speed-up.

For convenience, at time t, we define $w'(j_i)$ for jobs $j_i \neq j_\tau$ as follows: let $w'(j_i) = w(j_i)$ if WLAPS is processing j_i at time t, and let $w'(j_i) = 0$ otherwise. Note that $\sum_{i=1}^{n_a(t)} w'(j_i) = \beta w_{\mathrm{a}}(t)$. Intuitively, $w'(j_i)/\beta w_{\mathrm{a}}(t)$ is the fraction of processor speed received by j_i at time t. Specifically, job j_i runs at speed $\frac{w'(j_i)}{\beta w_{\mathrm{a}}(t)} s_{\mathrm{a}}(t)$.

Framework of analysis. To analyze WLAPS, we exploit amortization and potential functions to compare WLAPS with some offline algorithm OFF. As mentioned in Section 2, we need a potential function $\Phi(t)$ that satisfies the boundary, discrete-event and running conditions. Below we define a general form of $\Phi(t)$ that works for the traditional and arbitrary power models.

Potential function $\Phi(t)$. Consider any time t. Recall that $n_{\mathrm{a}}(t)$ and $w_{\mathrm{a}}(t)$ are respectively the number and total weight of active jobs in WLAPS. We define $n_{\mathrm{o}}(t)$ and $w_{\mathrm{o}}(t)$ similarly for OFF. We will drop the parameter t when t is clearly the current time. Define the coefficient of j_i, denoted c_i, to be $\sum_{k=1}^{i} w(j_k)$. Note that $c_{n_{\mathrm{a}}} = w_{\mathrm{a}}$. For any job j, let $q_{\mathrm{a}}(j,t)$ and $q_{\mathrm{o}}(j,t)$ be the remaining work of job j in WLAPS and OFF, respectively. For each j_i, let $x_i = \max\{q_{\mathrm{a}}(j_i,t) - q_{\mathrm{o}}(j_i,t), 0\}$ which is the amount of work of j_i in WLAPS that is lagging behind OFF. We say a job j_i is *lagging* if $x_i > 0$. Based on the notion of lagging, we define

$$\Phi(t) = \sum_{i=1}^{n_{\mathrm{a}}(t)} f(c_i) \cdot x_i \ ,$$

where $f(x)$ is some non-decreasing function of x (to be defined differently in the traditional and arbitrary power model).

We can check the boundary and discrete-event conditions without the detailed definition of $f(x)$. The boundary condition clearly holds due to the definition of Φ. Next, we check the discrete-event condition. When a job j arrives, j must be non-lagging and the coefficients of all existing jobs of WLAPS remain the same, so Φ does not change. When OFF completes a job, Φ does not change. When WLAPS completes a job, since $f(x)$ is non-decreasing and the coefficient of any other job either stays the same or decreases, Φ does not increase.

To show the running condition, we need to consider the two power models separately and have a different definition of the function f. Here we can only discuss some useful properties of Φ common to both power models. Consider any time t when Φ does not have discrete change. Let s_{a} and s_{o} be the current speeds of WLAPS and OFF, respectively. Among the jobs that WLAPS is processing at time t (i.e., $j_\tau, j_{\tau+1}, \ldots, j_{n_{\mathrm{a}}}$), our analysis will focus on those that are also lagging jobs. Denote the set of such lagging jobs as L. Note that $\sum_{i|j_i\in L} w'(j_i) \le \beta w_{\mathrm{a}}$. We further define another real number ϕ such that $\sum_{i|j_i\in L} w'(j_i) = \phi w_{\mathrm{a}}$. Note that $\phi \le \beta$. WLAPS is processing non-lagging jobs with total weight at least $(\beta-\phi)w_{\mathrm{a}}$, and these jobs are also active jobs for OPT. Thus, $w_{\mathrm{o}} \ge (\beta-\phi)w_{\mathrm{a}}$.

To bound the rate of change of Φ, we consider how Φ changes in an infinitesimal amount of time (from t to $t+\mathrm{d}t$), first due to WLAPS only (Lemma 1 (i)), and then due to OFF (Lemma 1 (ii)). We denote the rate of change of Φ due to WLAPS and OFF by $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t}$ and $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t}$, respectively. Note that $\frac{\mathrm{d}\Phi}{\mathrm{d}t} = \frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t} + \frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t}$.

Lemma 1. (i) $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t} \le -\frac{\phi}{\beta} \cdot f((1-\beta)w_{\mathrm{a}})s_{\mathrm{a}}$. (ii) $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le f(w_{\mathrm{a}}) \cdot s_{\mathrm{o}}$.

Proof. We first prove (i). It is trivial when $s_{\mathrm{a}} = 0$; it remains to consider $s_{\mathrm{a}} > 0$. By the definition of Φ, the execution of WLAPS can only decrease the potential. We will show that the rate of decrease is at least $\frac{\phi}{\beta} \cdot f((1-\beta)w_{\mathrm{a}})s_{\mathrm{a}}$, or equivalently, the rate of change is at most $-\frac{\phi}{\beta} \cdot f((1-\beta)w_{\mathrm{a}})s_{\mathrm{a}}$.

If we only consider the change due to WLAPS, for any $j_i \in L$ (WLAPS is processing j_i and j_i is lagging), its lagging size x_i is changing at the rate of $-\frac{w'(j_i)}{\beta w_{\mathrm{a}}} s_{\mathrm{a}}$. For other jobs $j_i \notin L$, x_i does not change. Note that as $f(x)$ is non-decreasing, we have $r \cdot f(c) \ge \int_{c-r}^{c} f(x)\,\mathrm{d}x$ for any constant $c \ge r$. Thus,

$$\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t} = \sum_{i|j_i\in L} \left(f(c_i) \cdot -\frac{w'(j_i)}{\beta w_{\mathrm{a}}} s_{\mathrm{a}}\right) = -\frac{s_{\mathrm{a}}}{\beta w_{\mathrm{a}}} \sum_{i|j_i\in L} f(c_i) w'(j_i) \le -\frac{s_{\mathrm{a}}}{\beta w_{\mathrm{a}}} \sum_{i|j_i\in L} \int_{c_i - w'(j_i)}^{c_i} f(x)\,\mathrm{d}x.$$

We view $\sum_{i|j_i\in L} \int_{c_i-w'(j_i)}^{c_i} f(x)\,\mathrm{d}x$ as the sum of integrations of some non-overlapping ranges. Since $f(x)$ is non-decreasing, the integration over a specific range must be at least the integration over a range of the same size and with smaller end-points, i.e., $\int_{c-r}^{c} f(x)\,\mathrm{d}x \ge \int_{c'-r}^{c'} f(x)\,\mathrm{d}x$ for $r \le c' \le c$. By the definition of coefficients, every job in L has a coefficient at least that of j_τ (i.e., c_τ). Furthermore, the total length of the integration ranges is $\sum_{i|j_i\in L} w'(j_i)$. By

"moving" the ranges towards the minimum possible endpoints starting from $c_\tau - w'(j_\tau)$, we have

$$\sum_{i|j_i \in L} \int_{c_i - w'(j_i)}^{c_i} f(x)\,\mathrm{d}x \geq \int_{c_\tau - w'(j_\tau)}^{c_\tau - w'(j_\tau) + \sum_{i|j_i \in L} w'(j_i)} f(x)\,\mathrm{d}x \text{ ; and}$$

$$\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \leq -\frac{s_\mathrm{a}}{\beta w_\mathrm{a}} \int_{c_\tau - w'(j_\tau)}^{c_\tau - w'(j_\tau) + \sum_{i|j_i \in L} w'(j_i)} f(x)\,\mathrm{d}x = -\frac{s_\mathrm{a}}{\beta w_\mathrm{a}} \int_{w_\mathrm{a} - \beta w_\mathrm{a} + \phi w_\mathrm{a}}^{w_\mathrm{a} - \beta w_\mathrm{a}} f(x)\,\mathrm{d}x.$$

Since $f(x)$ is non-decreasing, we have

$$\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \leq -\frac{s_\mathrm{a}}{\beta w_\mathrm{a}} \int_{(1-\beta)w_\mathrm{a}}^{(1-\beta+\phi)w_\mathrm{a}} f(x)\,\mathrm{d}x \leq -\frac{s_\mathrm{a}}{\beta w_\mathrm{a}} (\phi w_\mathrm{a} \cdot f((1-\beta)w_\mathrm{a})) = -\frac{\phi}{\beta} \cdot f((1-\beta)w_\mathrm{a}) s_\mathrm{a}\,.$$

For (ii), to upper bound $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t}$, the worst case is that OFF is processing the job j_{n_a} with the largest coefficient $c_{n_\mathrm{a}} = \sum_{k=1}^{n_a} w(j_k) = w_\mathrm{a}$, so $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \leq f(w_\mathrm{a}) \cdot s_\mathrm{o}$. □

In Sections 3.1 and 3.2, we will consider the two power models separately, and show the refinement of AJW and the definition of f that are sufficient to prove the running condition for each model.

3.1 Traditional Power Model

We consider the traditional power model, which assumes that the power function $P(s) = s^\alpha$ for some $\alpha > 1$. If the maximum speed T is unbounded, WLAPS follows exactly AJW and sets its speed at time t as $s_\mathrm{a}(t) = w_\mathrm{a}(t)^{1/\alpha}$. If T is bounded, we let WLAPS use a processor with maximum speed $(1+\epsilon)T$ for any $\epsilon > 0$, and set $s_\mathrm{a}(t) = \min(w_\mathrm{a}(t)^{1/\alpha}, (1+\epsilon)T)$.

Our main result is the following theorem.

Theorem 1. *Consider minimizing weighted flow plus energy.* (i) *If the maximum speed T is unbounded,* WLAPS *is $8\alpha^2$-competitive.* (ii) *If T is bounded,* WLAPS *is $\max(8\alpha^2, 8(1+\frac{1}{\epsilon})^2)$-competitive, using a processor with maximum speed $(1+\epsilon)T$.*

Unbounded maximum speed. To prove Theorem 1 (i), we set the offline algorithm OFF as the optimal offline algorithm OPT for minimizing weighted flow plus energy, and let $f(x) = x^{1-1/\alpha}$, which is clearly an non-decreasing function of x. Then the theorem follows from the running condition below.

Lemma 2. *Assume that $\beta = \frac{1}{2\alpha}$ and $\gamma = \frac{2}{\beta}$. At any time when Φ does not have discrete change, $\frac{\mathrm{d}G_\mathrm{a}}{\mathrm{d}t} + \gamma \frac{\mathrm{d}\Phi}{\mathrm{d}t} \leq 8\alpha^2 \cdot \frac{\mathrm{d}G_\mathrm{o}}{\mathrm{d}t}$.*

Proof. Note that $\frac{\mathrm{d}G_\mathrm{a}}{\mathrm{d}t} = w_\mathrm{a} + s_\mathrm{a}^\alpha = 2w_\mathrm{a}$, and $\frac{\mathrm{d}G_\mathrm{o}}{\mathrm{d}t} = w_\mathrm{o} + s_\mathrm{o}^\alpha \geq (\beta - \phi)w_\mathrm{a} + s_\mathrm{o}^\alpha$. By Lemma 1 (i), $\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \leq -\frac{\phi}{\beta} \cdot f((1-\beta)w_\mathrm{a}) s_\mathrm{a}$. Since $s_\mathrm{a} = w_\mathrm{a}^{1/\alpha}$ and $f(x) = x^{1-1/\alpha}$, we have $f(x)s_\mathrm{a} \geq x$ for any $0 \leq x \leq w_\mathrm{a}$. Thus, $\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \leq -\frac{\phi}{\beta} \cdot f((1-\beta)w_\mathrm{a}) s_\mathrm{a} \leq -\frac{\phi}{\beta}(1-\beta)w_\mathrm{a}$. On the other hand, by Lemma 1 (ii), $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \leq f(w_\mathrm{a}) \cdot s_\mathrm{o} = w_\mathrm{a}^{1-1/\alpha} s_\mathrm{o}$.

We apply the Young's Inequality [17][2], by setting $p = \alpha$, $q = \frac{\alpha}{\alpha-1}$, $x = s_{\mathrm{o}}$ and $y = w_{\mathrm{a}}^{1-1/\alpha}$. Then $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le w_{\mathrm{a}}^{1-1/\alpha} s_{\mathrm{o}} \le (1-\frac{1}{\alpha})w_{\mathrm{a}} + \frac{1}{\alpha}s_{\mathrm{o}}^{\alpha} = (1-2\beta)w_{\mathrm{a}} + \frac{1}{\alpha}s_{\mathrm{o}}^{\alpha}$.

Note that $\gamma = \frac{2}{\beta}$ and $\phi \le \beta$. Then

$$\begin{aligned}\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} &\le 2w_{\mathrm{a}} + \tfrac{2}{\beta}(1-2\beta)w_{\mathrm{a}} + \tfrac{2}{\beta\alpha}s_{\mathrm{o}}^{\alpha} - \tfrac{2\phi}{\beta^2}(1-\beta)w_{\mathrm{a}} \\ &= ((\tfrac{2}{\beta} - \tfrac{2\phi}{\beta^2}) + (-2 + \tfrac{2\phi}{\beta}))w_{\mathrm{a}} + \tfrac{2}{\beta\alpha}s_{\mathrm{o}}^{\alpha} \\ &\le \tfrac{2}{\beta^2}(\beta-\phi)w_{\mathrm{a}} + \tfrac{2}{\beta\alpha}s_{\mathrm{o}}^{\alpha} \le \tfrac{2}{\beta^2}w_{\mathrm{o}} + \tfrac{2}{\beta\alpha}s_{\mathrm{o}}^{\alpha} \le 8\alpha^2 \cdot \frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t}\ . \end{aligned}$$

□

Bounded maximum speed. To prove Theorem 1 (ii), we let OFF be the optimal offline algorithm OPT that uses a processor with maximum speed T, and let $f(x) = x/\min(x^{1/\alpha}, (1+\epsilon)T)$. Again, $f(x)$ is a non-decreasing function of x. We prove the running condition below.

Lemma 3. *Assume that* $\beta = \min(\frac{1}{2\alpha}, \frac{\epsilon}{2\epsilon+2})$ *and* $\gamma = \frac{2}{\beta}$. *At any time when* Φ *does not have discrete change,* $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \max(8\alpha^2, 8(1+\frac{1}{\epsilon})^2) \cdot \frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t}$.

Proof. We first show that $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le (1-2\beta)w_{\mathrm{a}} + \frac{1}{\alpha}s_{\mathrm{o}}^{\alpha}$. By Lemma 1 (ii), $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le f(w_{\mathrm{a}}) \cdot s_{\mathrm{o}}$. If $w_{\mathrm{a}}^{1/\alpha} \le (1+\epsilon)T$, we have $f(w_{\mathrm{a}}) = w_{\mathrm{a}}^{1-1/\alpha}$ and hence $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le w_{\mathrm{a}}^{1-1/\alpha}s_{\mathrm{o}}$. We can show that $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le (1-2\beta)w_{\mathrm{a}} + \frac{1}{\alpha}s_{\mathrm{o}}^{\alpha}$ in the same way as in Lemma 2. If $w_{\mathrm{a}}^{1/\alpha} > (1+\epsilon)T$, we have $f(w_{\mathrm{a}}) = \frac{w_{\mathrm{a}}}{(1+\epsilon)T}$. We use the fact that $s_{\mathrm{o}} \le T$ and $\frac{1}{1+\epsilon} = 1 - \frac{\epsilon}{1+\epsilon}$ to conclude that $\frac{\mathrm{d}\Phi_{\mathrm{o}}}{\mathrm{d}t} \le (1-\frac{\epsilon}{1+\epsilon})\frac{w_{\mathrm{a}}}{T}s_{\mathrm{o}} \le (1-\frac{\epsilon}{1+\epsilon})w_{\mathrm{a}} \le (1-2\beta)w_{\mathrm{a}}$.

Note that $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} = w_{\mathrm{a}} + (\min(w_{\mathrm{a}}^{1/\alpha}, (1+\epsilon)T))^{\alpha} \le 2w_{\mathrm{a}}$. Using the same argument as in Lemma 2, the lemma follows. □

3.2 Arbitrary Power Model

Consider an arbitrary power function $P(s)$. Recall that Q denotes the inverse of P. Let T be the maximum speed; if it does not exist, let $T = \infty$. We consider WLAPS being given a $(1+\epsilon)$-speedup processor for any $\epsilon > 0$, and define the speed of WLAPS at time t as $s_{\mathrm{a}}(t) = (1+\epsilon) \cdot \min(Q(w_{\mathrm{a}}(t)), T)$. By definition, the power required by WLAPS is $P(\min(Q(w_{\mathrm{a}}(t)), T))$.

We compare WLAPS with offline algorithms using a processor without speedup. Let OPT be an optimal offline algorithm. And let OFF be an optimal algorithm among all offline algorithms that scale its speed as $s_{\mathrm{o}}(t) = \min(Q(w_{\mathrm{o}}(t)), T)$, where $w_{\mathrm{o}}(t)$ is the total weight of active jobs at time t. Our main result is the following theorem.

Theorem 2. *For any* $\epsilon > 0$, *when* $\beta = \frac{\epsilon}{2\epsilon+2}$, WLAPS *with a* $(1+\epsilon)$*-speedup processor is* $8(1+\frac{1}{\epsilon})^2$*-competitive for weighted flow plus energy against* OFF.

In Section 4, we will show an offline algorithm which scales its speed as $s_{\mathrm{o}}(t) = \min(Q(w_{\mathrm{o}}(t)), T)$, and of which the total weighted flow plus energy incurred is at most two times of that of OPT (Corollary 2 (i)). Thus, OFF is a 2-approximation of OPT.

[2] Young's Inequality: For positive reals p, q, x, y where $\frac{1}{p} + \frac{1}{q} = 1$, $xy \le \frac{1}{p}x^p + \frac{1}{q}y^q$.

Corollary 1. *For any $\epsilon > 0$, when $\beta = \frac{\epsilon}{2\epsilon+2}$,* WLAPS *with a $(1+\epsilon)$-speedup processor is $16(1+\frac{1}{\epsilon})^2$-competitive for weighted flow plus energy.*

To prove Theorem 2, we set $f(x) = \frac{x}{\min(Q(x),T)}$,which is non-decreasing.[3] Then the theorem follows from the running condition below.

Lemma 4. *Assume that $\beta = \frac{\epsilon}{2\epsilon+2}$, and $\gamma = \frac{2}{\beta(1+\epsilon)}$. At any time when Φ does not have discrete change, $\frac{\mathrm{d}G_\mathrm{a}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 8(1+\frac{1}{\epsilon})^2 \cdot \frac{\mathrm{d}G_\mathrm{o}}{\mathrm{d}t}$.*

Proof. Note that $\frac{\mathrm{d}G_\mathrm{a}}{\mathrm{d}t} = w_\mathrm{a} + P(\min(Q(w_\mathrm{a}),T)) \le w_\mathrm{a} + P(Q(w_\mathrm{a})) = 2w_\mathrm{a}$ and $\frac{\mathrm{d}G_\mathrm{o}}{\mathrm{d}t} \ge w_\mathrm{o} \ge (\beta-\phi)w_\mathrm{a}$. By Lemma 1 (i), $\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \le -\frac{\phi}{\beta}\cdot f((1-\beta)w_\mathrm{a})s_\mathrm{a}$. Recall that $s_\mathrm{a} = (1+\epsilon)\min(Q(w_\mathrm{a}),T)$. By the definition of $f(x)$, we have $f(x)s_\mathrm{a} \ge (1+\epsilon)x$ for any $0 \le x \le w_\mathrm{a}$. Thus, $\frac{\mathrm{d}\Phi_\mathrm{a}}{\mathrm{d}t} \le -\frac{\phi}{\beta}\cdot f((1-\beta)w_\mathrm{a})s_\mathrm{a} \le -\frac{\phi}{\beta}(1+\epsilon)(1-\beta)w_\mathrm{a}$. On the other hand, by Lemma 1 (ii), $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \le f(w_\mathrm{a})\cdot s_\mathrm{o} = \frac{w_\mathrm{a}}{\min(Q(w_\mathrm{a}),T)}\cdot \min(Q(w_\mathrm{o}),T)$. If $w_\mathrm{a} \ge w_\mathrm{o}$, since Q is increasing, $Q(w_\mathrm{a}) \ge Q(w_\mathrm{o})$ and hence $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \le w_\mathrm{a}$. Otherwise, $w_\mathrm{a} < w_\mathrm{o}$. Since $f(x)$ is non-decreasing, $f(w_\mathrm{a}) \le f(w_\mathrm{o}) = \frac{w_\mathrm{o}}{\min(Q(w_\mathrm{o}),T)} = \frac{w_\mathrm{o}}{s_\mathrm{o}}$, so $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \le f(w_\mathrm{a})\cdot s_\mathrm{o} \le w_\mathrm{o}$. Therefore, $\frac{\mathrm{d}\Phi_\mathrm{o}}{\mathrm{d}t} \le \max(w_\mathrm{a}, w_\mathrm{o})$ and hence $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \max(w_\mathrm{a}, w_\mathrm{o}) - \frac{\phi}{\beta}(1+\epsilon)(1-\beta)w_\mathrm{a}$.

Recall that $\gamma = \frac{2}{\beta(1+\epsilon)}$, and note that $\frac{1}{1+\epsilon} = 1-2\beta$ and $\phi \le \beta \le 1$. Then

$$\begin{aligned}\frac{\mathrm{d}G_\mathrm{a}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} &\le 2w_\mathrm{a} + \tfrac{2(1-2\beta)}{\beta}\max(w_\mathrm{a}, w_\mathrm{o}) - \tfrac{2\phi}{\beta^2}(1-\beta)w_\mathrm{a}\\ &\le (\tfrac{2}{\beta} + (-2+\tfrac{2\phi}{\beta}))\max(w_\mathrm{a}, w_\mathrm{o}) - \tfrac{2\phi}{\beta^2}w_\mathrm{a} \le \tfrac{2}{\beta^2}(\beta\max(w_\mathrm{a}, w_\mathrm{o}) - \phi w_\mathrm{a})\\ &\le \tfrac{2}{\beta^2}\max((\beta-\phi)w_\mathrm{a}, \beta w_\mathrm{o}) \le \tfrac{2}{\beta^2}w_\mathrm{o} \le 8(1+\tfrac{1}{\epsilon})^2 \cdot \tfrac{\mathrm{d}G_\mathrm{o}}{\mathrm{d}t}\ .\end{aligned}$$

□

4 Clairvoyant Transformation to AJW

This section considers clairvoyant speed scaling using AJW under an arbitrary power function $P(s)$. Given any (online/offline) algorithm B that is using a $(1+\epsilon)$-speedup processor for $\epsilon \ge 0$ (if $\epsilon = 0$, there is no speedup), we give a clairvoyant algorithm LLB (latest lag behind) which simulates B and gives another schedule using the speed scaling policy AJW. We use LLB(B) to denote the resulting algorithm. LLB(B), when using a $(1+\epsilon)$-speedup processor as B, incurs at most twice the weighted flow plus energy of B (Theorem 3).

If we consider B to be an optimal offline algorithm OPT, we obtain a 2-approximate offline algorithm that scales the speed using AJW (Corollary 2 (i)). This offline result is needed in Section 3.2. For the online setting, recall that BPS is a competitive clairvoyant algorithm, and it scales the speed continuously over time. Using LLB, we effectively transform BPS to use discrete-time speed

[3] Since P is strictly increasing, Q is strictly increasing. Let ρ be a real such that $Q(\rho) = T$. For any $x \ge \rho$, $Q(x) \ge Q(\rho) = T$ and $f(x) = \frac{x}{\min(Q(x),T)} = \frac{x}{T}$ which is non-decreasing. For $x \in [0,\rho]$, $f(x) = \frac{x}{Q(x)}$. Note that P is strictly convex and Q is concave. For any $\lambda \in [0,1]$ and $x, y \in [0,\rho]$, $(1-\lambda)Q(x)+\lambda Q(y) \le Q((1-\lambda)x+\lambda y)$. Setting $x = 0$ gives $\lambda Q(y) \le Q(\lambda y)$ and hence $f(\lambda y) = \frac{\lambda y}{Q(\lambda y)} \le \frac{y}{Q(y)} = f(y)$.

scaling. Based on [6], one can show that for any $\epsilon > 0$, BPS when using a $(1+\epsilon)$-speedup processor is $2(1+\frac{1}{\epsilon})$-competitive. Thus, LLB(BPS) with a $(1+\epsilon)$-speedup processor is $4(1+\frac{1}{\epsilon})$-competitive (Corollary 2 (ii)). Note that LLB(BPS) has a better performance than WLAPS, yet the former only works clairvoyantly.

Algorithm LLB(B). Let T denote the maximum speed allowed by the power function $P(s)$ (if it is unbounded, then $T = \infty$). Consider any time t. Let $n_{\rm a}(t)$ and $w_{\rm a}(t)$ be respectively the number and total weight of active jobs in LLB. For any job j, let $q_{\rm a}(j,t)$ and $q_{\rm o}(j,t)$ be the remaining work of j at t in LLB and B, respectively. For each j_i, let $x_i = q_{\rm a}(j_i,t) - q_{\rm o}(j_i,t)$ be the difference in remaining work of LLB and B, and let y_i be the latest time that $x_i \le 0$. We say a job j_i is *lagging* if $x_i > 0$. We denote the active jobs in LLB as $j_1, j_2, \ldots, j_{n_{\rm a}(t)}$, which are arranged in increasing order of the time when they have become lagging; i.e., $y_1 \le y_2 \le \cdots \le y_{n_{\rm a}(t)}$ (ties are broken by job ids). Furthermore, let ℓ be the number of lagging jobs. Notice that for $i = 1$ to ℓ, $x_i > 0$ and $y_i < t$; and for $i = \ell + 1$ to $n_{\rm a}$, $x_i \le 0$ and $y_i = t$. Assume B is running at speed $s_{\rm o}$ at time t. LLB sets its speed $s_a = (1+\epsilon) \cdot \min(Q(w_{\rm a}(t)), T)$, and it targets the job j_a defined to be j_ℓ if $\ell > 0$, and $j_{n_{\rm a}}$ otherwise.

- If B is processing a job j_b with $x_b = 0$, then LLB runs j_b with speed $\min(s_{\rm a}, s_{\rm o})$ and runs j_a with the remaining speed $s_{\rm a} - \min(s_{\rm a}, s_{\rm o})$.
- Otherwise, (i.e. B is processing a job j_b with $x_b \neq 0$ or a job that is not active for LLB), it runs j_a with speed $s_{\rm a}$.

Our main result is Theorem 3 below, which implies Corollary 2.

Theorem 3. *Let* B *be an (online/offline) algorithm using a* $(1+\epsilon)$*-speedup processor where* $\epsilon \ge 0$*. Under an arbitrary power function,* LLB(B) *with a* $(1+\epsilon)$*-speedup processor incurs at most two times the total weighted flow plus energy of* B*.*

Corollary 2. *Consider minimizing weighted flow plus energy. Let* $w(t)$ *be the total weight of active jobs at time* t *and let* T *be the maximum speed.*

(i) *Let* OPT *be the optimal offline algorithm without using any speedup.* LLB (OPT) *is a 2-approximate (offline) algorithm that scales the speed at any time* t *as* $\min(Q(w(t)), T)$*.*

(ii) *For any* $\epsilon > 0$*,* LLB(BPS) *using a* $(1+\epsilon)$*-speedup processor is a* $4(1+\frac{1}{\epsilon})$*-competitive online algorithm that scales the speed at any time* t *as* $(1+\epsilon) \cdot \min(Q(w(t)), T)$*.*

Before proving Theorem 3, we have the following observation. Both LLB and B are using a $(1+\epsilon)$-speedup processor (note that when $\epsilon = 0$, the processor has no speedup). Such a processor, when given power $P(s)$, runs at speed $(1+\epsilon)s$. In other words, the power function of a $(1+\epsilon)$-speedup processor is $\bar{P}(s) = P(\frac{s}{1+\epsilon})$. We let $\bar{T} = (1+\epsilon)T$ which is the maximum speed of the $(1+\epsilon)$-speedup processor. Like P, the power function $\bar{P}$ satisfies that $\bar{P}(0) = 0$, and $\bar{P}$ is defined, strictly increasing, strictly convex, continuous and differentiable at all speeds in $[0, \bar{T}]$; if $\bar{T} = \infty$, the speed range is $[0, \infty)$. We let $\bar{Q}$ denote the inverse of $\bar{P}$. At any

time t, the speed of LLB is $s_{\mathrm{a}}(t) = (1+\epsilon)\cdot\min(Q(w_{\mathrm{a}}(t)), T) = \min(\bar{Q}(w_{\mathrm{a}}(t)), \bar{T})$. Thus, it suffices to analyze LLB based on the power function $\bar{P}$.

To show Theorem 3, we again exploit amortization and potential functions as shown in Section 2. We derive a potential function Φ that satisfies the three required conditions. Consider any time t. Recall that the active jobs in LLB are denoted as $j_1, j_2, \ldots, j_{n_{\mathrm{a}}(t)}$. Define the coefficient c_i of j_i to be $\sum_{k=1}^{i} w(j_k)$. The potential function $\Phi(t)$ is defined as follows.

$$\Phi(t) = \sum\nolimits_{i=1}^{n_{\mathrm{a}}(t)} f(c_i)\cdot\max(x_i, 0) \quad \text{where} \quad f(x) = \bar{P}'(\bar{Q}(x)).$$

Note that $\bar{P}'$ is the first derivative of $\bar{P}$. Since $\bar{P}$ is convex, $\bar{P}'$ is non-increasing, which together with that $\bar{Q}(x)$ is non-decreasing, implies that $\bar{P}'(\bar{Q}(x))$ is also non-decreasing. Therefore, $f(x)$ is a non-decreasing function of x.

We can show that such potential function Φ satisfies the boundary and discrete-event conditions. More interestingly, by the definition of LLB, we can show that at any time, $\bar{Q}(c_\ell) \le \bar{T}$, where c_ℓ is the coefficient of job j_ℓ and also the total weight of lagging jobs in LLB. This allows us to prove the running condition that at any time when Φ does not have discrete change, $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + 2\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 2\frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t}$. The detailed proof is given in the full version of the paper. Then Theorem 3 follows.

References

1. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4), 49 (2007)
2. Andrew, L., Wierman, A., Tang, A.: Optimal speed scaling under arbitrary power functions. ACM SIGMETRICS Performance Evaluation Review 37(2), 39–41 (2009)
3. Brooks, D.M., Bose, P., Schuster, S.E., Jacobson, H., Kudva, P.N., Buyuktosunoglu, A., Wellman, J.D., Zyuban, V., Gupta, M., Cook, P.W.: Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. IEEE Micro 20(6), 26–44 (2000)
4. Bansal, N., Chan, H.L.: Weighted flow time does not admit $O(1)$-competitive algorithms. In: Proc. SODA, pp. 1238–1244 (2009)
5. Bansal, N., Chan, H.L., Lam, T.W., Lee, L.K.: Scheduling for speed bounded processors. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 409–420. Springer, Heidelberg (2008)
6. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proc. SODA, pp. 693–701 (2009)
7. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. SIAM Journal on Computing 39(4), 1294–1308 (2009)
8. Chan, H.L., Edmonds, J., Lam, T.W., Lee, L.K., Marchetti-Spaccamela, A., Pruhs, K.: Nonclairvoyant speed scaling for flow and energy. In: Proc. STACS, pp. 255–264 (2009)
9. Chan, H.L., Edmonds, J., Pruhs, K.: Speed scaling of processes with arbitrary speedup curves on a multiprocessor. In: Proc. SPAA, pp. 1–10 (2009)
10. Chan, S.H., Lam, T.W., Lee, L.K., Ting, H.F., Zhang, P.: Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors. In: Proc. CATS, pp. 3–10 (2010)

11. Kalyanasundaram, B., Pruhs, K.: Minimizing flow time nonclairvoyantly. Journal of the ACM 50(4), 551–567 (2003)
12. Lam, T.W., Lee, L.K., Ting, H.F., To, I., Wong, P.: Sleep with guilt and work faster to minimize flow plus energy. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 665–676. Springer, Heidelberg (2009)
13. Lam, T.W., Lee, L.K., To, I., Wong, P.: Competitive non-migratory scheduling for flow time and energy. In: Proc. SPAA, pp. 256–264 (2008)
14. Lam, T.W., Lee, L.K., To, I., Wong, P.: Speed scaling functions for flow time scheduling based on active job count. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 647–659. Springer, Heidelberg (2008)
15. Motwani, R., Phillips, S., Torng, E.: Nonclairvoyant scheduling. Theoretical Computer Science 130(1), 17–47 (1994)
16. Mudge, T.: Power: A first-class architectural design constraint. IEEE Computer 34(4), 52–58 (2001)
17. Steele, J.M.: The Cauchy-Schwarz master class: An introduction to the art of mathematical inequalities, p. 136. Cambridge University Press, Cambridge (2004)
18. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Proc. FOCS, pp. 374–382 (1995)