Speed Scaling Functions for Flow Time Scheduling based on Active Job Count

Tak-Wah Lam[1], Lap-Kei Lee[1], Isaac K. K. To[2], and Prudence W. H. Wong[2]

[1] Department of Computer Science, University of Hong Kong.
{twlam, lklee}@cs.hku.hk
[2] Department of Computer Science, University of Liverpool.
{isaacto, pwong}@liverpool.ac.uk

Abstract. We study online scheduling to minimize flow time plus energy usage in the dynamic speed scaling model. We devise new speed scaling functions that depend on the number of active jobs, replacing the existing speed scaling functions in the literature that depend on the remaining work of active jobs. The new speed functions are more stable and also more efficient. They can support better job selection strategies to improve the competitive ratios of existing algorithms [5,8], and, more importantly, to remove the requirement of extra speed. These functions further distinguish themselves from others as they can readily be used in the non-clairvoyant model (where the size of a job is only known when the job finishes). As a first step, we study the scheduling of batched jobs (i.e., jobs with the same release time) in the non-clairvoyant model and present the first competitive algorithm for minimizing flow time plus energy (as well as for weighted flow time plus energy); the performance is close to optimal.

1 Introduction

Energy usage is an important concern in recent research on online scheduling. A popular approach to reducing energy usage is *dynamic speed scaling* (see e.g., [10, 13]), which allows a processor to vary its speed dynamically. A processor incurs an energy of s^α per unit time when running at speed s, where $\alpha \geq 2$ (typically 2 or 3 [10, 20]). Running a job slower saves energy, yet it takes longer and may affect performance. A lot of effort has been devoted to revisiting classical scheduling problems with dynamic speed scaling and energy concern taken into consideration (see [14] for a survey). The challenge arises from the conflicting objectives of providing good "quality of service" (QoS) and conserving energy. These studies first focused on the *infinite speed model* [24] where any speed may be used, and have recently shifted to the more realistic *bounded speed model* [12], which imposes a bound T on the maximum allowable speed.

One commonly used QoS measure for job scheduling is the total flow time. Here, jobs with arbitrary size are released at unpredictable times and the flow

* This research is partly supported by Hong Kong RGC Grant HKU/7140/06E and EPSRC Grant EP/E028276/1.

	$\alpha = 2$	$\alpha = 3$
Infinite speed model ($T = \infty$)	5.236 [8] **2.667** [this paper]	7.940 [8] **3.252** [this paper]
Bounded speed model	10.472 with max speed $1.618T$ [5] **3.6** with max speed T [this paper]	11.910 with max speed $1.466T$ [5] **4** with max speed T [this paper]

Table 1. Results on clairvoyant scheduling for minimizing flow time plus energy. Note that the new results in this paper do not demand extra speed.

time of a job is the time elapsed since it arrives until it is completed. Preemption is allowed without penalty. When energy is not a concern, the objective is simply to minimize the total flow time of all jobs, and the online algorithm SRPT (shortest remaining processing time) is optimal [3]. If jobs carry weights, another online algorithm HDF (highest density first) has been shown to be $(1 + 1/\epsilon)$-competitive for weighted flow time, when using $(1+\epsilon)$ times faster processor [9]; in fact, no online algorithm can be $O(1)$-competitive without extra speed [4].

The above results assume clairvoyance, i.e., the size of a job is known at release time. In some applications like operating systems, job size is only known when the job finishes. This is referred to as the non-clairvoyant model. The earlier work focused on batched jobs (i.e., all jobs have the same release time), and Motwani et al. [19] have shown that the online algorithm Round Robin is 2-competitive for flow time. There is a matching lower bound of 2 when the number of jobs is large. Kim and Chwa [17] further showed that a weighted version of Round Robin is also 2-competitive for weighted flow time. For jobs with arbitrary release times, Kalyanasundaram and Pruhs [16] showed that SETF (shortest elapsed time first) is $(1+\epsilon)$-speed $(1+1/\epsilon)$-competitive for flow time. The result has also been generalized to weighted flow time with the same performance [6,17].

Flow time and energy. To understand the tradeoff between flow time and energy, Albers and Fujiwara [1] proposed combining the dual objectives into a single objective of minimizing the sum of total flow time and energy. The intuition is that, from an economic viewpoint, users are willing to pay a certain (say, x) units of energy to reduce one unit of flow time. By changing the units of time and energy, one can further assume that $x = 1$ and thus would like to optimize total flow time plus energy. Albers and Fujiwara focused on the clairvoyant model and their work has been improved by Bansal et al. [5, 8]. As far as we know, there is no previous work on the non-clairvoyant model.

In the infinite speed model, Bansal et al. [8] gave an $O(1)$-competitive online algorithm for minimizing flow time plus energy; precisely, the competitive ratio is $\mu_\epsilon\gamma_1$, where ϵ is any positive constant, $\mu_\epsilon = \max\{(1 + 1/\epsilon), (1 + \epsilon)^\alpha\}$ and $\gamma_1 = \max\{2, \frac{2(\alpha-1)}{\alpha-(\alpha-1)^{1-1/(\alpha-1)}}\}$. E.g., if $\alpha = 2$, the competitive ratio is 5.236. Very recently, Bansal et al. [5] adapted this result to the bounded speed model. Assuming that the online algorithm can have a higher maximum speed of $(1+\epsilon)T$ for any $\epsilon > 0$, the competitive ratio in this case increases slightly to $\mu_\epsilon\gamma_2$, where $\gamma_2 = 2\alpha/(\alpha-(\alpha-1)^{1-1/(\alpha-1)}) = (2+o(1))\alpha/\ln\alpha$. Table 1 shows the competitive ratios for some fixed α. Both results also hold for weighted flow time plus energy.

Follow-up questions. The speed function of the algorithms by Bansal et al. [5,8] depends on the *remaining work* of "active" jobs (i.e., jobs that have not been completed). There are several questions related to such a speed function.

- In the non-clairvoyant model, work-based speed functions are not applicable since job size, and hence remaining work, is not known to the online algorithm. Can one devise a speed function for the non-clairvoyant model?
- Work-based speed functions would demand the processor speed to change continuously which is undesirable practically. Can one design a more stable speed function that changes in a discrete manner?
- The algorithms in [5, 8] require extra speed even in the unweighted setting. In contrast, the classic result on flow-time scheduling does not need extra speed [3]. This is perhaps due to the inefficiency of the work-based speed functions; specifically, they are sometimes slower than a critical threshold ($(\frac{n}{\alpha-1})^{1/\alpha}$ where n is the number of active jobs). When this happens, we can decrease the flow time plus energy by increasing the speed. It is natural to ask whether a speed function that never goes below this threshold can work without extra speed and gives a better competitive ratio.

Our contribution. In this paper, we answer affirmatively the above questions by introducing new speed functions that depend on the number of active jobs. They are more stable, changing only at job arrival or completion. Using these functions leads to improvements in the clairvoyant model and provides the first competitive results on non-clairvoyant scheduling of batched jobs.

- **Clairvoyant scheduling:** Define the speed function AJC to be $n^{1/\alpha}$, where n is the number of active jobs. We use SRPT (instead of SJF (shortest job first) in [5,8]) to select jobs. Our algorithm is more competitive for minimizing flow time plus energy and does not need extra speed: for the infinite and bounded speed model, the competitive ratios are respectively $\beta_1 = 2/(1 - \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$ and $\beta_2 = 2(\alpha+1)/(\alpha - \frac{\alpha-1}{(\alpha+1)^{1/(\alpha-1)}})$. Table 1 compares these ratios with those in [5,8]. The improvement is more significant for large α, as β_1 and β_2 tend to $2\alpha/\ln\alpha$, while $\mu_\epsilon\gamma_1$ and $\mu_\epsilon\gamma_2$ [5,8] tend to $2(\alpha/\ln\alpha)^2$.
- **Non-clairvoyant scheduling:** We focus on batched jobs, i.e., when all jobs are released at time 0. For scheduling unweighted jobs, we use the speed function $\text{AJC}^* = (\frac{n}{\alpha-1})^{1/\alpha}$, where n is the total number of active jobs. Coupling with Round Robin, we give an algorithm that is $(2-\frac{1}{\alpha})$-competitive in the infinite speed model and 2-competitive in the bounded speed model. The latter inherits a lower bound of 2 from flow-time scheduling [19]. We can further generalize this algorithm for minimizing weighted flow time plus energy, and the corresponding ratios become $(2-\frac{1}{\alpha})^2$ and 4, respectively.

Technically speaking, the analysis of existing clairvoyant algorithms requires indirect comparison via a notion called fractional flow. In contrast, we divide the time into "stable intervals", and directly compare the flow time of the online algorithm against an optimal offline algorithm in each interval. This makes

the analysis tighter. For the non-clairvoyant algorithms, we first prove that the performance of our algorithm is close to that of a clairvoyant algorithm based on SJF (shortest job first) plus AJC*. Then we show that SJF-AJC* is optimal against any offline algorithm for minimizing flow time plus energy.

Remarks. The theoretical study of energy-efficient scheduling was initiated by Yao, Demers and Shenker [24]. They considered deadline scheduling in the infinite speed model. Their result was improved by Bansal et al. [7], and extended to the bounded speed model by Chan et al. [12] and Bansal et al. [5]. Irani et al. [15] considered a setting where the processor has a sleep state. Pruhs et al. [21] also studied offline scheduling for minimizing the total flow time on a processor with a given amount of energy. The offline problem of minimizing the makespan subject to a fixed amount of energy has been studied in [11, 22]. Recently, multiprocessor setting has received attention, with interesting offline [11, 22] and online results [2, 18]. As some of these multiprocessor algorithms (in particular, [18]) are based on the single-processor algorithms in [5,8], our new result on clairvoyant scheduling immediately leads to an improvement.

2 Definitions and Notations

We consider a job set $\mathcal{J}$ to be scheduled on a processor whose speed can be varied between 0 and T, the speed bound. In the infinite speed model, $T = \infty$, so any speed is allowed. When running at speed s, the processor processes s units of work and consumes s^α units of energy in each unit of time, where $\alpha \geq 2$. Preemption is allowed; a preempted job can resume at the point of preemption. We use r_i and p_i to denote the release time and work requirement (or *size*) of a job J_i in $\mathcal{J}$, respectively. If $r_i = 0$ for all jobs, we call $\mathcal{J}$ *batched jobs*.

Consider a particular time t in a schedule of $\mathcal{J}$. For any job J_i in $\mathcal{J}$, we let q_i denote its remaining work at t, and say it is *active* if $r_i \leq t$ and $q_i > 0$. The *flow time* F_i of J_i is the time elapsed since it arrives until it is completed. The *total flow time* is given by $F = \sum_i F_i$. Note that $F = \int_0^\infty n(t)\,\mathrm{d}t$, where $n(t)$ denotes the number of active jobs at time t, and the energy consumption is $E = \int_0^\infty s(t)^\alpha\,\mathrm{d}t$, where $s(t)$ is the speed of the processor at time t. Our aim is to minimize the total flow time plus energy, $G = F + E$. We observe that each of E, F and G is the integration over all time from 0 to ∞. We call the integration over a period of time to be the *contribution* to their respective value during that period. E.g., the contribution to G from t_1 to t_2 is $\int_{t_1}^{t_2}(n(t) + s(t)^\alpha)\,\mathrm{d}t$.

3 Clairvoyant Scheduling with Arbitrary Release Times

In this section, we study the case where a job arrives at arbitrary time and its size is known upon arrival. Define the speed function AJC (active job count) to be $\min\{T, n^{1/\alpha}\}$, where n is the number of active jobs and T is maximum speed. Coupling with SRPT, we have the algorithm **SRPT-AJC**. We first explain that SRPT gives the best job selection (Lemma 1). In the next two subsections, we analyze SRPT-AJC in the infinite speed and bounded speed models.

Lemma 1. *Consider a job sequence $\mathcal{J}$. Suppose a schedule S for $\mathcal{J}$ uses speed function f. Then, among all schedules of $\mathcal{J}$ using the speed function f, the one selecting jobs in accordance with SRPT incurs the least total flow time.*

Proof. We modify S in multiple steps to a SRPT schedule. In each step the new schedule S' has total flow time reduced. Then the lemma follows.

Let t be the first time when S does not follow SRPT, running J_i instead of the shortest remaining work job J_j. S' differs from S during the time intervals since t when S runs either job: J_j is run to completion before J_i, using the same speed function. J_j thus completes in S' earlier than J_i does in S. So the sum of their completion time, and thus the total flow time, is less in S' than in S. □

3.1 Analysis of SRPT-AJC for the Infinite Speed Model

We analyze SRPT-AJC for the infinite speed model, comparing it against an optimal offline schedule OPT. By Lemma 1, OPT uses the SRPT policy as well.

Overview. Consider any time t. Let $G_{\mathrm{a}}(t)$ and $G_{\mathrm{o}}(t)$ be the flow time plus energy incurred up to t by SRPT-AJC and OPT, respectively. We drop the parameter t when it refers to the current time. Our analysis exploits amortization and potential functions (e.g., [8, 12]): if there is a potential function $\Phi(t)$ and a value β such that the followings hold, then SRPT-AJC is β-competitive.

- *Boundary condition:* Φ is initially 0 and finally non-negative.
- *Arrival condition:* When a job is released, Φ does not increase.
- *Running condition:* At any other time, the rate of change of G_{a} plus that of Φ is no more than β times the rate of change of G_{o}, i.e., $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \beta \frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t}$.

We first show such a potential function Φ, whose design is motivated by the work in [5, 8]. We will choose β to be $2/(1 - \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$. Then we show that the above conditions hold, and thus can conclude the competitiveness of SRPT-AJC.

Theorem 1. *SRPT-AJC is β_1-competitive for flow time plus energy in the infinite speed model, where $\beta_1 = 2/(1 - \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$.*

Potential function $\Phi(t)$. Consider any time t. For any $q \ge 0$, let $n_{\mathrm{a}}(q)$ denote the current number of active jobs with remaining work at least q in SRPT-AJC, and similarly $n_{\mathrm{o}}(q)$ for OPT. So $n_{\mathrm{a}}(0)$ and $n_{\mathrm{o}}(0)$ are the total number of active jobs in the schedules, which we abbreviate as n_{a} and n_{o}, respectively. It is useful to consider $n_{\mathrm{a}}(q)$ and $n_{\mathrm{o}}(q)$ as functions of q which change when a job arrives or runs for a while (see Fig. 3.1). We define the potential function as

$$\Phi(t) = \eta \int_0^\infty \phi(q)\,\mathrm{d}q \ \text{, where } \phi(q) = \left(\sum_{i=1}^{n_{\mathrm{a}}(q)} i^{1-1/\alpha} \right) - n_{\mathrm{a}}(q)^{1-1/\alpha} n_{\mathrm{o}}(q) \ ,$$

and $\eta = 2/(1 - \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$. (In bounded speed model, $\eta = 2/(1 - \frac{1-1/\alpha}{(\alpha+1)^{1/(\alpha-1)}})$.)

At $t = 0$ or ∞, no job remains, so $\Phi = 0$. The integration of the first term of $\phi(q)$ is proportional to the total flow time plus energy of SRPT-AJC after t if no more job arrives (which is $2 \int_0^\infty \sum_{i=1}^{n_{\mathrm{a}}(q)} i^{1-1/\alpha}\,\mathrm{d}q$), paying the cost once OPT completes all jobs. The second term of $\phi(q)$ makes the arrival condition hold.

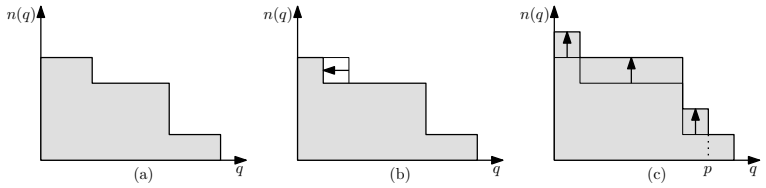

Fig. 1. (a) At any time, $n_{\mathrm{a}}(q)$ or $n_{\mathrm{o}}(q)$ (denoted by $n(q)$ above) is a step function containing unit height stripes, the area under $n(q)$ is the total remaining work. (b) If we run a job selected with SRPT at speed s for a period of time Δ, the top stripe shrinks by $s\Delta$. (c) When a job of size p is released, $n(q)$ increases by 1 for all $q \le p$.

Lemma 2. *When a job arrives, the change of Φ is non-positive.*

Proof. Suppose a job J_j arrives. For $q > p_j$, $n_{\mathrm{a}}(q)$ and $n_{\mathrm{o}}(q)$, and hence $\phi(q)$, are unchanged. For $q \le p_j$, both $n_{\mathrm{a}}(q)$ and $n_{\mathrm{o}}(q)$ increase by 1 (see Fig. 3.1(c)). So the first term of $\phi(q)$ increases by $(n_{\mathrm{a}}(q)+1)^{1-1/\alpha}$. The increase of the term $n_{\mathrm{a}}(q)^{1-1/\alpha} n_{\mathrm{o}}(q)$ is interpreted in two-steps: (i) increase to $(n_{\mathrm{a}}(q)+1)^{1-1/\alpha} n_{\mathrm{o}}(q)$, and (ii) increase to $(n_{\mathrm{a}}(q)+1)^{1-1/\alpha}(n_{\mathrm{o}}(q)+1)$. The increase in step (ii), $(n_{\mathrm{a}}(q)+1)^{1-1/\alpha}$, covers the increase in the first term of $\phi(q)$, so $\phi(q)$ cannot increase. □

Running condition. The rest of this section proves the following lemma.

Lemma 3. *When no job arrives, $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \beta_1 \frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t}$ where $\beta_1 = 2/(1-\frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$.*

Consider any time t. Let s_{a} and s_{o} be the speed of SRPT-AJC and OPT, respectively, and q_{a} and q_{o} be the remaining work of the job they run. We divide the time line into *stable intervals* by breaking it when the following events occur.

- A job arrives, or is completed by either SRPT-AJC or OPT.
- The speed s_{a} or the job chosen by SRPT-AJC changes.
- The speed s_{o} or the job chosen by OPT changes.
- Either $n_{\mathrm{o}}(q_{\mathrm{a}})$ or $n_{\mathrm{a}}(q_{\mathrm{o}})$ changes.

Φ does not increase on job arrival (Lemma 2), and is unchanged on other events above ($\phi(q)$ changes for single q). So we focus on its change in a stable interval.

We first bound $\frac{\mathrm{d}\Phi}{\mathrm{d}t}$. Consider a stable interval of length $\mathrm{d}t$. We analyze $\mathrm{d}\Phi$ as if $\phi(q)$ is modified in two steps: (i) $n_{\mathrm{a}}(q)$ decreases due to the execution of SRPT-AJC. (ii) $n_{\mathrm{o}}(q)$ decreases due to the execution of OPT. We denote these changes by $\mathrm{d}\Phi_1$ and $\mathrm{d}\Phi_2$, respectively. Then $\mathrm{d}\Phi = \mathrm{d}\Phi_1 + \mathrm{d}\Phi_2$.

Lemma 4. *Consider a stable interval of length $\mathrm{d}t$. (i) $\mathrm{d}\Phi_1 \le \eta(n_{\mathrm{o}} - n_{\mathrm{a}})\mathrm{d}t$; (ii) $\mathrm{d}\Phi_2 \le \eta n_{\mathrm{a}}^{1-1/\alpha} s_{\mathrm{o}} \mathrm{d}t$.*

Proof. (i) Consider SRPT-AJC. $n_{\mathrm{a}}(q)$ decreases from n_{a} to $n_{\mathrm{a}} - 1$ for all $q \in [q_{\mathrm{a}} - s_{\mathrm{a}}\mathrm{d}t, q_{\mathrm{a}}]$ (see Fig. 3.1(b)). For any $q \in [q_{\mathrm{a}} - s_{\mathrm{a}}\mathrm{d}t, q_{\mathrm{a}}]$, $\phi(q)$ is changed by:

$$\left(\sum_{i=1}^{n_{\mathrm{a}}-1} i^{1-1/\alpha}\right) - (n_{\mathrm{a}}-1)^{1-1/\alpha} n_{\mathrm{o}}(q) - \left(\sum_{i=1}^{n_{\mathrm{a}}} i^{1-1/\alpha}\right) + n_{\mathrm{a}}^{1-1/\alpha} n_{\mathrm{o}}(q)$$
$$\le -n_{\mathrm{a}}^{1-1/\alpha} + n_{\mathrm{a}}^{-1/\alpha} n_{\mathrm{o}}(q) \le -n_{\mathrm{a}}^{1-1/\alpha} + n_{\mathrm{a}}^{-1/\alpha} n_{\mathrm{o}} \ .$$

(The first inequality is due to $n^{1-1/\alpha} - (n-1)^{1-1/\alpha} \leq n^{-1/\alpha}$ for any $n \geq 1$. The second holds since $n_{\mathrm{o}}(q) \leq n_{\mathrm{o}}$.) Recall that $s_{\mathrm{a}} = n_{\mathrm{a}}^{1/\alpha}$. Integrating over all q,

$$\mathrm{d}\Phi_1 \leq \eta(-n_{\mathrm{a}}^{1-1/\alpha} + n_{\mathrm{a}}^{-1/\alpha} n_{\mathrm{o}})(n_{\mathrm{a}}^{1/\alpha}\mathrm{d}t) = \eta(n_{\mathrm{o}} - n_{\mathrm{a}})\mathrm{d}t \ .$$

(ii) Consider OPT. As in (i), $\phi(q)$ is changed only for $q \in [q_{\mathrm{o}} - s_{\mathrm{o}}\mathrm{d}t, q_{\mathrm{o}}]$. Note that $n_{\mathrm{a}}(q) \leq n_{\mathrm{a}}$ for all q. Then for any $q \in [q_{\mathrm{o}} - s_{\mathrm{o}}\mathrm{d}t, q_{\mathrm{o}}]$, $\phi(q)$ is changed by

$$-n_{\mathrm{a}}(q)^{1-1/\alpha}(n_{\mathrm{o}}(q) - 1) + n_{\mathrm{a}}(q)^{1-1/\alpha} n_{\mathrm{o}}(q) = n_{\mathrm{a}}(q)^{1-1/\alpha} \leq n_{\mathrm{a}}^{1-1/\alpha} \ .$$

Integrating over all q, we have $\mathrm{d}\Phi_2 \leq \eta n_{\mathrm{a}}^{1-1/\alpha} s_{\mathrm{o}}\mathrm{d}t$. □

Proof (Lemma 3). We introduce a constant $\mu > 0$ into Lemma 4 (ii) using the Young's Inequality (see Corollary 9 of [8]), leading to $\mathrm{d}\Phi_2 \leq \eta n_{\mathrm{a}}^{1-1/\alpha} s_{\mathrm{o}}\mathrm{d}t \leq \big(\eta(1 - \frac{1}{\alpha})\mu^{\alpha/(\alpha-1)} n_{\mathrm{a}} + \frac{\eta s_{\mathrm{o}}^{\alpha}}{\alpha\mu^{\alpha}}\big)\mathrm{d}t$. Since $s_{\mathrm{a}} = n_{\mathrm{a}}^{1/\alpha}$, $\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} = n_{\mathrm{a}} + s_{\mathrm{a}}^{\alpha} = 2n_{\mathrm{a}}$. Also, $\frac{\mathrm{d}G_{\mathrm{o}}}{\mathrm{d}t} = n_{\mathrm{o}} + s_{\mathrm{o}}^{\alpha}$. Since $\mathrm{d}\Phi = \mathrm{d}\Phi_1 + \mathrm{d}\Phi_2$, by Lemma 4 (i), we have

$$\frac{\mathrm{d}G_{\mathrm{a}}}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \leq 2n_{\mathrm{a}} + \eta\Big(n_{\mathrm{o}} - n_{\mathrm{a}} + \big(1 - \frac{1}{\alpha}\big)\mu^{\alpha/(\alpha-1)} n_{\mathrm{a}} + \frac{s_{\mathrm{o}}^{\alpha}}{\alpha\mu^{\alpha}}\Big) \ . \qquad (1)$$

Lemma 3 follows with $\beta_1 = \eta = 2/(1 - \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}})$ and $\mu = \alpha^{-1/\alpha}$. □

3.2 Analysis of SRPT-AJC for the Bounded Speed Model

We now turn to the bounded speed model. We use the same potential function while setting $\eta = 2/(1 - \frac{1-1/\alpha}{(\alpha+1)^{1/(\alpha-1)}})$. Most of the analysis in Sect. 3.1 still holds, except Lemma 4 (i). As the speed used might be T instead of $n^{1/\alpha}$ if n is large, the decrease in Φ_1 is not sufficient to guarantee the original competitive ratio. We observe that the difference $(n_{\mathrm{a}} - n_{\mathrm{o}})$ is upper bounded by T^{α} (Lemma 5). Then the competitiveness is only slightly worse (Theorem 2).

Lemma 5. *At any time, the number of active jobs in SRPT-AJC and OPT satisfy* $n_{\mathrm{a}} - n_{\mathrm{o}} \leq T^{\alpha}$.

Proof. This is trivial if $n_{\mathrm{a}} < T^{\alpha}$. Suppose $n_{\mathrm{a}} \geq T^{\alpha}$ at time t. Let t_0 be the last moment before t when n_{a} changes from less than T^{α} to at least T^{α}. Consider the interval $[t_0, t)$, where SRPT-AJC runs at full speed. Suppose k jobs arrive, and OPT completes x of them, so $n_{\mathrm{o}} \geq k - x$. Since SRPT maximizes the number of jobs completed at any time [23], SRPT-AJC also completes at least x jobs. So n_{a} increases by at most $k - x \leq n_{\mathrm{o}}$ since t_0, and the lemma follows. □

Theorem 2. *SRPT-AJC is* β_2*-competitive for flow time plus energy in the bounded speed model, where* $\beta_2 = 2(\alpha+1)/(\alpha - \frac{\alpha-1}{(\alpha+1)^{1/(\alpha-1)}})$.

Proof. The boundary and arrival conditions still hold. It remains to establish the running condition. Its analysis is split into two cases.

Case 1: $n_a \le T^\alpha$. The arguments in Sect. 3.1 still hold, leading to (1). Set $\mu = (\alpha+1)^{-1/\alpha}$ and recall that $\eta = 2/(1 - \frac{1-1/\alpha}{(\alpha+1)^{1/(\alpha-1)}})$, (1) implies $\frac{\mathrm{d}G_a}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \eta n_o + (1+1/\alpha)\eta s_o^\alpha \le \beta_2 \frac{\mathrm{d}G_o}{\mathrm{d}t}$.

Case 2: $n_a > T^\alpha$. The arguments in Sect. 3.1 hold, except Lemma 4 (i). We still have $\frac{\mathrm{d}G_o}{\mathrm{d}t} = n_o + s_o^\alpha$ and $\frac{\mathrm{d}G_a}{\mathrm{d}t} = n_a + T^\alpha < 2n_a$. For $\mathrm{d}\Phi_1$,

$$\mathrm{d}\Phi_1 \le \eta(-n_a^{1-1/\alpha} + n_a^{-1/\alpha} n_o) T \mathrm{d}t = \eta(n_o - n_a)\Big(\frac{T^\alpha}{n_a}\Big)^{1/\alpha} \mathrm{d}t \ .$$

If $n_o \ge n_a$, $\mathrm{d}\Phi_1 \le \eta(n_o - n_a)\mathrm{d}t$ as before (since $T^\alpha/n_a \le 1$), leading to (1), so the arguments in Case 1 apply. If $n_o < n_a$, Lemma 5 implies

$$\mathrm{d}\Phi_1 \le -\eta(n_a - n_o)\Big(\frac{T^\alpha}{n_a}\Big)^{1/\alpha} \mathrm{d}t \le \frac{-\eta(n_a - n_o)^{1+1/\alpha}}{n_a^{1/\alpha}} \mathrm{d}t \ .$$

Note that $f(n_a - n_o) \ge f(n_a) - f'(n_a)\, n_o$ for the convex function $f(x) = x^{1+1/\alpha}$, which is equivalent to $(n_a - n_o)^{1+1/\alpha} \ge n_a^{1+1/\alpha} - (1+1/\alpha) n_a^{1/\alpha} n_o$. We thus have

$$\mathrm{d}\Phi_1 \le -\eta\big((n_a^{1+1/\alpha} - (1+1/\alpha) n_a^{1/\alpha} n_o)/n_a^{1/\alpha}\big)\mathrm{d}t = \eta\big((1+1/\alpha)n_o - n_a\big)\mathrm{d}t \ ,$$

$$\text{and } \frac{\mathrm{d}G_a}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 2n_a + \eta\Big((1+\frac{1}{\alpha})n_o - n_a + \big(1 - \frac{1}{\alpha}\big)\mu^{\alpha/(\alpha-1)} n_a + \frac{s_o^\alpha}{\alpha\mu^\alpha}\Big) \ .$$

Setting $\mu = (\alpha+1)^{-1/\alpha}$, this implies $\frac{\mathrm{d}G_a}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le (1+\frac{1}{\alpha})\eta(n_o + s_o^\alpha) = \beta_2 \frac{\mathrm{d}G_o}{\mathrm{d}t}$. □

4 Non-clairvoyant Scheduling of Batched Jobs

In this section, we study non-clairvoyant scheduling of batched jobs. Recall that the online algorithm knows the size of a job only when the job finishes.

AJC* and RR-AJC*. For batched jobs, we use a more energy-conservative speed function AJC*, defined as $\min\{T, (\frac{n(t)}{\alpha-1})^{1/\alpha}\}$, where $n(t)$ is the number of active jobs at t. To cope with unknown job sizes, RR-AJC* uses AJC* with RR: split the processor equally among all active jobs.

To analyze RR-AJC*, we consider a clairvoyant algorithm SJF-AJC* (shortest job first plus AJC*). In Sect. 4.1, we show an interesting relation that the flow time plus energy incurred by RR-AJC* is close to that of SJF-AJC*. In Sect. 4.2 we complete the analysis by showing SJF-AJC* is optimal.

In some applications, jobs may carry weights to reflect their importance. We use w_i to denote the weight of a job J_i, and define its *density* ρ_i to be w_i/p_i. The weighted flow time of a job is simply its flow time multiplied by its weight. The above result can be generalized for weighted flow time plus energy as follows.

AJW* and WRR-AJW*. The speed function AJW* (active job weight) is defined as $\min\{T, (\frac{w(t)}{\alpha-1})^{1/\alpha}\}$ where $w(t)$ is the total weight of active jobs at t. The algorithm WRR-AJW* uses AJW* with the WRR policy: split the processor among all active jobs in the ratio of their weights. We define the *normalized work* of a job as its work divided by its weight. Every active job has the same normalized processed work at any time.

To analyze WRR-AJW*, we consider a clairvoyant algorithm HDF-AJW* (highest density first plus AJW*). The relation between WRR-AJW* and HDF-AJW* is the same as before (Sect. 4.1). However, HDF-AJW* is not optimal. Yet we show that HDF-AJW* is competitive in Sect. 4.3.

We use the weighted setting as a common platform, where RR-AJC* (resp. SJF-AJC*) is WRR-AJW* (resp. HDF-AJW*) with job weights all equal to one. Without loss of generality, we assume the input $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is in increasing job density order, i.e., $\rho_1 \leq \rho_2 \leq \cdots \leq \rho_n$ (ties are broken by job IDs).

4.1 Comparing WRR-AJW* against HDF-AJW*

As jobs are batched, WRR implies that jobs complete in the order of normalized work, from J_n to J_1. Thus $w(t) = \sum_{j=1}^{i} w_j$ if J_i is the smallest normalized work job at t. We can thus compare WRR-AJW* against HDF-AJW* easily.

Lemma 6. *For scheduling batched jobs, the weighted flow time plus energy of WRR-AJW* is (i) at most $(2 - 1/\alpha)$ times of HDF-AJW* in the infinite speed model, and (ii) at most 2 times of HDF-AJW* in the bounded speed model.*

To prove Lemma 6, we focus on the contributions to weighted flow time and energy during the time when a particular job J_i is the job with the smallest normalized work in WRR-AJW*. Due to the WRR policy, each of the remaining jobs J_j with $1 \leq j \leq i$ is run for the same amount of normalized work. We thus evaluate the weighted flow time and energy incurred by WRR-AJW* during this period, denoted as $F_{\mathrm{w}}(J_i)$ and $E_{\mathrm{w}}(J_i)$. They are compared against the weighted flow time and energy incurred by HDF-AJW* to process that same amount of normalized work for each of these jobs, denoted as $F_{\mathrm{h}}(J_i)$ and $E_{\mathrm{h}}(J_i)$. We show that both $F_{\mathrm{w}}(J_i)/F_{\mathrm{h}}(J_i)$ and $E_{\mathrm{w}}(J_i)/E_{\mathrm{h}}(J_i)$ are no greater than $(2 - 1/\alpha)$ for $T = \infty$, and 2 for general T. Summing over all J_i leads to Lemma 6.

Lemma 7. *If $T = \infty$, $F_{\mathrm{w}}(J_i) \leq (2-1/\alpha)F_{\mathrm{h}}(J_i)$ and $E_{\mathrm{w}}(J_i) \leq (2-1/\alpha)E_{\mathrm{h}}(J_i)$.*

Proof. Due to the rules in the speed function AJW*, when $T = \infty$ we have $F_{\mathrm{w}}(J_i) = (\alpha - 1)E_{\mathrm{w}}(J_i)$ and $F_{\mathrm{h}}(J_i) = (\alpha - 1)E_{\mathrm{h}}(J_i)$. So it suffices to show $F_{\mathrm{w}}(J_i) \leq (2 - 1/\alpha)F_{\mathrm{h}}(J_i)$.

Let us first consider WRR-AJW*. Let Δh be the normalized work processed for each job $J_1, \ldots, J_i$. We have seen that $w(t) = \sum_{k=1}^{i} w_k$, $s = (\frac{w(t)}{\alpha-1})^{1/\alpha}$. The amount of work processed for J_j $(1 \leq j \leq i)$ is $w_j \Delta h$, so the total weighted flow time incurred $F_{\mathrm{w}}(J_i) = \sum_{j=1}^{i} w_j \Delta h \sum_{k=1}^{i} w_k / s = (\alpha-1)^{1/\alpha}(\sum_{j=1}^{i} w_j)^{2-1/\alpha}\Delta h$.

In contrast, HDF-AJW* runs only the highest density (i.e., least normalized work) job. At time t when processing J_j, $w(t) = \sum_{k=1}^{j} w_k$, $s = (\frac{w(t)}{\alpha-1})^{1/\alpha}$. The weighted flow time incurred for processing an amount of work $w_j \Delta h$ for J_j is thus $w_j \Delta h(\sum_{k=1}^{j} w_k)/s = (\alpha - 1)^{1/\alpha}\Delta h(\sum_{k=1}^{j} w_k)^{1-1/\alpha} w_j$. We thus have

$$F_{\mathrm{h}}(J_i) = (\alpha - 1)^{1/\alpha}\Delta h \sum_{j=1}^{i} \Big(\sum_{k=1}^{j} w_k\Big)^{1-1/\alpha} w_j \ .$$

To approximate $\sum_{j=1}^{i} \big(\sum_{k=1}^{j} w_k\big)^{1-1/\alpha} w_j$, we use the staircase-like function

$$f(x) = (\textstyle\sum_{k=1}^{j} w_k)^{1-1/\alpha} \qquad \text{if } x \in [\sum_{k=1}^{j-1} w_k, \sum_{k=1}^{j} w_k) \text{ where } 1 \le j \le i \quad .$$

Note that $\sum_{j=1}^{i}(\sum_{k=1}^{j} w_k)^{1-1/\alpha} w_j$ is exactly $\int_0^{\sum_{j=1}^{i} w_j} f(x)\,\mathrm{d}x$. On the other hand, $f(x) \ge x^{1-1/\alpha}$ for all $x \in [0, \sum_{j=1}^{i} w_j)$. We thus have

$$\sum_{j=1}^{i}\Big(\sum_{k=1}^{j} w_k\Big)^{1-1/\alpha} w_j \ge \int_0^{\sum_{j=1}^{i} w_j} x^{1-1/\alpha}\,\mathrm{d}x = \frac{(\sum_{j=1}^{i} w_j)^{2-1/\alpha}}{2-1/\alpha} \ ,$$

and $F_{\mathrm{h}}(J_i) \ge (\alpha-1)^{1/\alpha}\Delta h\big(\sum_{j=1}^{i} w_j\big)^{2-1/\alpha}/(2-\frac{1}{\alpha}) = F_{\mathrm{w}}(J_i)/(2-\frac{1}{\alpha})$. □

Proof (Lemma 6). We obtain (i) ($T = \infty$) by summing the relations about F_{h} and F_{w} and those about E_{h} and E_{w} in Lemma 7 over all J_i.

For (ii) ($T < \infty$), suppose WRR-AJW* processes Δh normalized work for each of $J_1, \ldots, J_i$. We first focus on $E_{\mathrm{w}}(J_i)$ and $E_{\mathrm{h}}(J_i)$. If $(\sum_{j=1}^{i} w_j/(\alpha-1))^{1/\alpha} \le T$, Lemma 7 applies, so $E_{\mathrm{w}}(J_i) \le (2-1/\alpha)E_{\mathrm{h}}(J_i)$. Otherwise, we try to find a $k \in \{1, \ldots, i\}$ such that $\sum_{j=1}^{k} w_j$ is exactly $(\alpha-1)T^{\alpha}$ (so that the speed bound is just not exceeded). If no such k exists, for the sake of analysis we split some job J_u into two jobs J_{u1} and J_{u2} with the same density and total weight, so that $w_{u1} + \sum_{j=1}^{u-1} w_j = (\alpha-1)T^{\alpha}$. We set $k = u1$. This job splitting does not affect the speed function, and thus the energy consumption, of either WRR-AJW* and HDF-AJW* (speed used for J_{u1}, J_{u2} and J_u are all T). We now notice that both WRR-AJW* and HDF-AJW* run $J_{k+1}, \ldots, J_i$ at speed T, consuming the same energy E_0. The other jobs are run as if $T = \infty$, so Lemma 7 leads to $E_{\mathrm{w}}(J_i) - E_0 \le (2-1/\alpha)(E_{\mathrm{h}}(J_i) - E_0)$. This implies $E_{\mathrm{w}}(J_i) \le (2-1/\alpha)E_{\mathrm{h}}(J_i)$.

We now compare $F_{\mathrm{w}}(J_i)$ and $F_{\mathrm{h}}(J_i)$. Again the interesting case is when $(\sum_{j=1}^{i} w_j/(\alpha-1))^{1/\alpha} > T$, otherwise $F_{\mathrm{w}}(J_i) \le (2-1/\alpha)F_{\mathrm{h}}(J_i)$ by Lemma 7. For WRR-AJW*, the processor uses speed T, so the time needed is $\sum_{j=1}^{i} w_j\Delta h/T$, and $F_{\mathrm{w}}(J_i) = (\sum_{j=1}^{i} w_j)^2\Delta h/T$. For HDF-AJW*, the time needed to run J_j for $w_j\Delta h$ units of work is at least $w_j\Delta h/T$ (since the speed used cannot be faster than T), incurring weighted flow time of at least $\sum_{k=1}^{j} w_k w_j\Delta h/T$. We thus have $F_{\mathrm{h}}(J_i) \ge \sum_{j=1}^{i}\sum_{k=1}^{j} w_k w_j\Delta h/T > (\sum_{j=1}^{i} w_j)^2\Delta h/2T = F_{\mathrm{w}}(J_i)/2$.

Summing these relations over all J_i gives the desired ratio in Lemma 6. □

4.2 Analysis of SJF-AJC*

We show that the speed function AJC* minimizes the flow time plus energy for scheduling batched jobs using SJF (equivalently, SRPT) (Lemma 8), implying the optimality of SJF-AJC* for flow time plus energy. Combining with Lemma 6, we obtain the competitive ratio of RR-AJC* (Theorem 3).

Lemma 8. *Consider a set of batched jobs $\mathcal{J}$. Among all schedules of $\mathcal{J}$ using SJF for job selection, the schedule that incurs the minimum flow time plus energy sets the speed at any time t as $\min\{T, (\frac{n(t)}{\alpha-1})^{1/\alpha}\}$, where $n(t)$ is the number of active jobs at t.*

Proof. Consider a particular job J_i in the optimal schedule. We only need to consider cases where its speed is constant, otherwise we can average the speed to reduce energy usage without affecting flow time. Note that $n(t)$ is unchanged when J_i is run. Suppose J_i runs at speed s. Then its contribution to flow time plus energy is $n(t)p_i/s + s^{\alpha-1}p_i$, which is minimized when $s = (\frac{n(t)}{\alpha-1})^{1/\alpha}$. □

Theorem 3. *For scheduling batched jobs to minimize flow time plus energy, the algorithm RR-AJC* is (i) $(2-1/\alpha)$-competitive in the infinite speed model, and (ii) 2-competitive in the bounded speed model.*

Proof. Since every other schedule can be modified to the SJF-AJC* schedule by Lemma 1 and then Lemma 8 while reducing total flow time plus energy, SJF-AJC* is optimal for total flow time plus energy. The theorem thus follows naturally from Lemma 6. □

4.3 Analysis of HDF-AJW*

When jobs are weighted, the clairvoyant algorithm HDF-AJW* is not optimal for weighted flow time plus energy. Instead we analyze HDF-AJW* via a variant concerning fractional flow. Due to space limitation, we only sketch the ideas.

Consider a schedule of $\mathcal{J}$. At any time t, the *fractional weight* $\widehat{w}_i$ of J_i is defined to be $w_i(q_i/p_i)$ (recall that q_i is the remaining work of J_i). We define $\widehat{w}(t) = \sum_{J_i \text{ is active}} \widehat{w}_i$. The *fractional flow* is defined as $\widehat{F} = \int_0^\infty \widehat{w}(t)\,\mathrm{d}t$. We now define our variant of HDF-AJW*.

Algorithm HDF-FW. It differs from HDF-AJW* only in the speed function, which uses the speed function FW defined as $(\frac{\widehat{w}(t)}{\alpha-1})^{1/\alpha}$ at any time t.

We follow the framework in Sect. 4.2 to show the optimality of HDF-FW: Firstly, HDF minimizes fractional flow for a fixed speed function. Secondly, FW minimizes the fractional flow plus energy for scheduling batched jobs using HDF.

Lemma 9. *Consider a set of weighted batched jobs. The algorithm HDF-FW is optimal for minimizing the fractional flow plus energy of the schedule.*

By comparing the contribution to the weighted flow time plus energy of HDF-AJW* against that to the fractional flow plus energy of HDF-FW when each job J_i is running, we can show the following lemma.

Lemma 10. *Consider a set of batched jobs. Let $\widehat{G}_{\mathrm{hf}}$ be the fractional flow plus energy with HDF-FW. Then the weighted flow time plus energy with HDF-AJW* is (i) $G_{\mathrm{h}} \le (2-1/\alpha)\widehat{G}_{\mathrm{hf}}$ in the infinite speed model, and (ii) $G_{\mathrm{h}} \le 2\widehat{G}_{\mathrm{hf}}$ in the bounded speed model.*

We note that for any schedule, the fractional flow is at most the weighted flow time. Thus HDF-AJW* is $(2-1/\alpha)$-competitive for weighted flow time plus energy. Together with Lemma 6, we obtain the competitive ratio of WRR-AJW*.

Theorem 4. *Consider a set of weighted batched jobs. The algorithm WRR-AJW* is (i) $(2-1/\alpha)^2$-competitive in the infinite speed model, and (ii) 4-competitive in the bounded speed model.*

References

1. S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Alg.*, 3(4):49, 2007.
2. S. Albers, F. Muller, and S. Schmelzer. Speed scaling on parallel processors. In *Proc. SPAA*, pages 289–298, 2007.
3. K. R. Baker. *Introduction to Sequencing and Scheduling.* Wiley, New York, 1974.
4. N. Bansal and H. L. Chan. Weighted flow time does not have $O(1)$ competitive algorithms. Manuscript.
5. N. Bansal, H. L. Chan, T. W. Lam, and L. K. Lee. Scheduling for speed bounded processors. In *Proc. ICALP*, 2008. To appear.
6. N. Bansal and K. Dhamdhere. Minimizing weighted flow time. *ACM Trans. Alg.*, 3(4):39, 2007.
7. N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.
8. N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. In *Proc. SODA*, pages 805–813, 2007.
9. L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs. Online weighted flow time and deadline scheduling. *J. Discrete Algorithms*, 4(3):339–352, 2006.
10. D. Brooks, P. Bose, S. Schuster, H. M. Jacobson, P. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.
11. D. P. Bunde. Power-aware scheduling for makespan and flow. In *Proc. SPAA*, pages 190–196, 2006.
12. H. L. Chan, W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak, and P. W. H. Wong. Energy efficient online deadline scheduling. In *Proc. SODA*, 2007.
13. D. Grunwald, P. Levis, C. B. Morrey, and M. Neufeld. Policies for dynamic clock scheduling. In *Proc. OSDI*, pages 73–86, 2000.
14. S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 32(2):63–76, 2005.
15. S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Alg.*, 3(4):41, 2007.
16. B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.
17. J.-H. Kim and K.-Y. Chwa. Non-clairvoyant scheduling for weighted flow time. *IPL*, 87(1):31–37, 2003.
18. T. W. Lam, L. K. Lee, I. K. K. To, and P. W. H. Wong. Competitive non-migratory scheduling for flow time and energy. In *Proc. SPAA*, pages 256–264, 2008.
19. R. Motwani, S. Phillips, and E. Torng. Nonclairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.
20. T. N. Mudge. Power: A first-class architectural design constraint. *IEEE Comp.*, 34(4):52–58, 2001.
21. K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. In *Proc. SWAT*, pages 14–25, 2004.
22. K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. In *Proc. WAOA*, pages 307–319, 2005.
23. L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.
24. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. FOCS*, pages 374–382, 1995.