Scheduling for Weighted Flow Time and Energy with Rejection Penalty*

Sze-Hang Chan[1], Tak-Wah Lam[2], and Lap-Kei Lee[3]

1,2 Department of Computer Science, University of Hong Kong, Hong Kong.
{shchan, twlam}@cs.hku.hk

3 Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany.
lklee@mpi-inf.mpg.de

Abstract

This paper revisits the online problem of flow-time scheduling on a single processor when jobs can be rejected at some penalty [4]. The *user cost* of a job is defined as the weighted flow time of the job plus the penalty if it is rejected before completion. For jobs with arbitrary weights and arbitrary penalties, Bansal et al. [4] gave an online algorithm that is $O((\log W + \log C)^2)$-competitive for minimizing the total user cost when using a slightly faster processor, where W and C are the max-min ratios of job weights and job penalties, respectively. In this paper we improve this result with a new algorithm that can achieve a constant competitive ratio independent of W and C when using a slightly faster processor. Note that the above results assume a processor running at a fixed speed. This paper shows more interesting results on extending the above study to the dynamic speed scaling model, where the processor can vary the speed dynamically and the rate of energy consumption is a cubic or any increasing function of speed. A scheduling algorithm has to control job admission and determine the order and speed of job execution. This paper studies the tradeoff between the above-mentioned user cost and energy, and it shows two O(1)-competitive algorithms and a lower bound result on minimizing the user cost plus energy. These algorithms can also be regarded as a generalization of the recent work on minimizing flow time plus energy when all jobs must be completed (see the survey paper [1]).

1998 ACM Subject Classification F.2.2[Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems—Sequencing and scheduling

Keywords and phrases Online scheduling, weighted flow time, rejection penalty, speed scaling

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.392

1 Introduction

It is not uncommon that a server rejects some jobs (in particular, low-priority jobs) during peak load, yet it is non-trivial how to strike a balance between the cost due to longer response time and the cost of rejecting some jobs. Bansal et al. [4] initiated the study of flow-time scheduling on a single processor when jobs can be rejected at some penalty. Specifically, jobs are released online with arbitrary sizes, weights and penalties. Consider a schedule which may reject some jobs before completion, each job defines a *user cost* equal to its weighted flow time plus the penalty if it is rejected, where the flow time is the time elapsed since a job is released until it is completed or rejected. In this penalty model, the scheduler aims at minimizing the total user cost of all jobs.

* This research was mainly done when the first two authors were visiting MPI, whose hospitality was greatly appreciated.

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 392–403

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Assuming jobs have uniform penalty and unit weight, Bansal et al. [4] gave an online algorithm that is 2-competitive for minimizing the total user cost. For jobs with arbitrary penalties and arbitrary weights, they give a resource augmentation result which achieves a competitive ratio of $O((\log W + \log C)^2)$ when using a slightly faster processor (precisely, $(1+\epsilon)$-speed processor for any $\epsilon > 0$), where W is the max-min ratio of job weights and C is the max-min ratio of job penalties. They also show a lower bound of $\Omega(\max(n^{\frac{1}{4}}, C^{\frac{1}{2}}))$ without using a faster processor, where n is the number of jobs in the job sequence. Note that for the special case when each job has infinite penalty, no jobs would be rejected and the problem reduces to the classic problem of minimizing weighted flow time only. In this case, Becchetti et al. [8] showed a better resource augmentation result, achieving $O(1)$-competitiveness for weighted flow time when using $(1+\epsilon)$-speed processor.

In this paper, we extend the results on rejection penalty [4] in two different directions. First of all, we improve the upper bound result on arbitrary penalties and arbitrary weights. Our online algorithm is constant competitive when using a $(1+\epsilon)$-speed processor, where the constant does not depend on W and C. In other words, for the special case when jobs must be all completed (with infinite penalty), our new algorithm has a comparable performance (but with a larger constant) as Becchetti et al's algorithm [8].

All the above results assume the processor running at a fixed speed. The main results in this paper are on extending the above study of rejection penalty to the dynamic speed scaling model [15] and taking energy into consideration. Specifically, it is assumed that the processor can vary its speed dynamically between 0 and some maximum speed T, and the power P increases with the speed s according to a certain function, say, $P(s) = s^3$. In this setting, a scheduling algorithm has to control job admission and determine the order and speed of job execution, and we are interested to measure the user cost as well as the total energy usage. Note that minimizing user cost and minimizing energy are orthogonal objectives. In this paper, we consider the problem of minimizing a linear combination of the user cost and energy, or simply the user cost plus energy. This problem can also be considered as a generalization of the existing work on minimizing weighted flow time plus energy where job rejection is not allowed [2, 9, 6, 13, 7, 3] (see related work below).

Speed scaling results. For jobs with uniform penalty and unit weight, we give a 6-competitive algorithm for minimizing the user cost plus energy. This algorithm ensures that the penalty of rejected jobs is always at most the flow time plus energy incurred thus far. Intuitively, it maintains a good balance between the flow time plus energy and the penalty. Next, we consider jobs with arbitrary penalties. We show a lower bound result illustrating the problem of minimizing the user cost plus energy being fundamentally more difficult than that of flow time plus energy. Specifically, we assume that $P(s) = s^\alpha$ for some $\alpha > 1$ and jobs have unit weight, and we show that any online algorithm has a competitive ratio of $\Omega(\alpha^{1/2-\epsilon})$ where ϵ is arbitrarily small, even if the maximum speed T is unbounded. This lower bound implies that the competitive ratio must grow with the steepness of the power function (α), while the problem of minimizing flow time plus energy admits a 2-competitive algorithm for any arbitrary power function [7, 3]. We turn to resource augmentation and consider giving the online algorithm a more energy-efficient processor which, using 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, based on which we devise an online algorithm for the arbitrary-penalty and arbitrary-weight setting. For any power function $P(s)$, our new algorithm is $O(1)$-competitive for minimizing the user cost plus energy when using a $(1+\epsilon)$-speedup processor. This algorithm, unlike the first one, rejects jobs only at their arrival time and therefore never wastes energy on rejected jobs.

Amortization and potential functions have become standard tool for analyzing algorithms

for minimizing flow time plus energy (e.g., [9, 13, 7, 3]). When jobs can be rejected, the online algorithm A and the optimal offline algorithm OPT may have completed two different sets of jobs. This complicates the analysis. For the case of uniform penalty and unit weight, the potential analysis only allows us to bound the flow time plus energy incurred by some special active jobs in A in terms of the cost of OPT. For cost incurred by other active jobs, our technique is an accounting argument to upper bound this cost of A by the total penalty of OPT. For arbitrary penalty and arbitrary weight jobs, taking the advantage of the use of speedup processor, we can directly incorporate the job penalty into the potential analysis and the maximum speed constraint T into the potential function.

Related work on dynamic speed scaling. To reduce energy usage, major chip manufacturers like Intel and IBM are now producing processors that can support *dynamic speed scaling*, which allows operating systems to manage the power by scaling the processor speed dynamically. How to exploit speed scaling effectively has become an interesting problem for the algorithmic community. Yao et al. [15] were the first to consider online job scheduling that takes speed scaling and energy usage into consideration. They considered a model where a processor can vary its speed s, and the energy is consumed at the rate s^α for some constant $\alpha > 1$ (in CMOS based processors, α is believed to be 3 [5]). Running jobs slower saves energy, yet it takes longer time. The challenge arises from the conflicting objectives of optimizing energy usage and some quality of service such as flow time. To understand their tradeoff, Albers and Fujiwara [2] initiated the study of minimizing a linear combination of the total 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. Following Albers and Fujiwara's work, there is a chain of work on speed scaling algorithms [2, 9, 6, 13, 7, 3], gradually improving the competitive ratios as well as dropping the assumptions on the speed-to-power functions. Now the best known algorithms can work for any arbitrary power function. For jobs with unit weight, a 2-competitive algorithm has been obtained [3]. For arbitrary weight, a competitive ratio of $O(1 + \frac{1}{\epsilon})$ can be achieved using a $(1 + \epsilon)$-speedup processor [7, 11].

Power functions and notations. Throughout the paper, we assume $P(0) = 0$, and P is defined, strictly increasing, strictly convex, continuous and differentiable at all speeds in $[0, T]$; if $T = \infty$, 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). We use Q to denote P^{-1}. Note that Q is strictly increasing and concave. E.g., if $P(s) = s^\alpha$, then $Q(x) = x^{1/\alpha}$. For each job j, we use $p(j)$, $w(j)$ and $v(j)$ to denote its work, weight and penalty.

Organization of the paper. The following discussion focuses on the results on the dynamic speed scaling model only. Our improved result on minimizing the user cost alone on a fixed-speed processor would be shown as a special case in Section 3. Section 2 considers jobs with uniform penalty and unit weight and presents a 6-competitive algorithm for minimizing the user cost plus energy. Section 3 gives the results on jobs with arbitrary penalties and arbitrary weights. Finally, a lower bound result is given in Section 4.

2 Uniform Penalty and Unit Weight

This section considers jobs with the same penalty $c > 0$ and unit weight. We give an online algorithm UPUW for minimizing flow plus penalty plus energy. The job rejection policy of UPUW is similar to that in [4], but the involvement of speed scaling and energy complicates

the analysis and demands a potential function. Our main result is the following theorem.

▶ **Theorem 1.** *Consider jobs with uniform penalty and unit weight. Algorithm* UPUW *is 6-competitive for minimizing flow plus penalty plus energy.*

Algorithm UPUW. At time t, let $n_{\mathrm{a}}(t)$ and $s_{\mathrm{a}}(t)$ be respectively the number of active jobs (i.e., jobs that have been released but not yet finished) and the speed of UPUW. Recall that Q is the inverse of the power function P. We set $s_{\mathrm{a}}(t) = \min(Q(n_{\mathrm{a}}(t)+1), T)$. UPUW always runs the job with the smallest remaining work (SRPT) at speed $s_{\mathrm{a}}(t)$ (ties are broken by job ids). Let ϕ be a counter that counts the flow plus energy incurred until time t, i.e., $\phi(t) = \int_0^t (n_{\mathrm{a}}(x) + P(s_{\mathrm{a}}(x)))\,\mathrm{d}x$. Whenever ϕ crosses a multiple of c, UPUW rejects the active job with the largest remaining work (if $n_{\mathrm{a}}(t) > 0$).

To prove Theorem 1, we compare UPUW with the optimal offline schedule OPT. Consider any job sequence. Let G_{a} be the total flow plus energy of UPUW and R_{a} be the total penalty of UPUW, and similarly define G_{o} and R_{o} for OPT. Let t_e be the time when all jobs are completed or rejected by both UPUW and OPT. By definition of UPUW, $G_{\mathrm{a}} = \phi(t_e)$ and $R_{\mathrm{a}} \le G_{\mathrm{a}}$. Thus, we have

▶ **Fact 2.** *The flow plus penalty plus energy of* UPUW *is* $G_{\mathrm{a}} + R_{\mathrm{a}} \le 2\phi(t_e)$.

To upper bound $\phi(t_e)$, we define another counter ψ such that at any time t, $\psi(t) \ge \phi(t)$. Then it suffices to upper bound $\psi(t_e)$ by the cost of OPT. We define ψ as follows: Initially, $\psi(0) = 0$. Whenever OPT rejects a job at t, ψ increases by c. At other times, if $\psi = \phi$, ψ increases at the same rate as ϕ, else (i.e., $\psi > \phi$), ψ stays the same.

Analysis framework. We will upper bound $\psi(t_e)$ in terms of G_{o} and R_{o}. Note that $\psi(t)$ is non-decreasing and increases in two cases: (Case 1) ψ increases by c whenever OPT rejects a job, and (Case 2) ψ increases at the same rate as ϕ whenever $\psi(t) = \phi(t)$. The increase due to Case 1 is bounded by R_{o}. To bound the increase due to Case 2, at any time t, we define a special subset of active jobs, denoted $B(t)$, as follows. Let $k(t) = \lfloor \frac{\psi(t)}{c} \rfloor - \lfloor \frac{\phi(t)}{c} \rfloor$. Let $B(t)$ be the set of the $n_{\mathrm{a}}(t) - k(t)$ active jobs in UPUW with the smallest remaining work ($B(t) = \emptyset$ if $n_{\mathrm{a}}(t) < k(t)$), and let $\tilde{n}_{\mathrm{a}}(t)$ denote the size of $B(t)$. Whenever $\psi(t) = \phi(t)$, $k(t) = 0$. If $\tilde{n}_{\mathrm{a}}(t) = 0$, it implies $n_{\mathrm{a}}(t) \le 0$, and $\phi(t)$ as well as $\psi(t)$ do not increase. Thus, the increase of ψ due to Case 2 is bounded by the flow plus energy incurred by UPUW during times when $\tilde{n}_{\mathrm{a}}(t) \ge 1$, which is upper bounded in Lemma 3. To prove Lemma 3, we follow the analysis in [7, 3] but adapt the potential function to focus only on jobs in $B(t)$ instead of all active jobs in UPUW. Like [7, 3], the potential analysis requires an upper bound on $\tilde{n}_{\mathrm{a}}(t) - n_{\mathrm{o}}(t)$ at any time t, where $n_{\mathrm{o}}(t)$ is the number of active jobs in OPT (Lemma 5). Yet with job rejections, we need new technique to obtain such upper bound, which will be proven via a mapping of jobs in $B(t)$ to jobs in a schedule related to OPT (Lemma 6).

We first state Lemma 3 and show how this lemma leads to Theorem 1. For any time interval I, let $G_{\mathrm{a}}[I]$ and $G_{\mathrm{o}}[I]$ be the flow plus energy incurred during I by UPUW and OPT, respectively.

▶ **Lemma 3.** *For any time interval* $I = (t_1, t_2)$ *such that* $\tilde{n}_{\mathrm{a}}(t_1) = \tilde{n}_{\mathrm{a}}(t_2) = 0$ *and* $\tilde{n}_{\mathrm{a}}(t) > 0$ *for any* $t \in I$, $G_{\mathrm{a}}[I] \le 3 \cdot G_{\mathrm{o}}[I] + 2\int_{t\in I} k(t)\,\mathrm{d}t$.

Proof of Theorem 1. Let $I_1, I_2, ..., I_m$ be all the intervals in $[0, t_e]$ such that for each $I_i = (t_1, t_2)$, $\tilde{n}_{\mathrm{a}}(t_1) = \tilde{n}_{\mathrm{a}}(t_2) = 0$ and $\tilde{n}_{\mathrm{a}}(t) > 0$ for any $t \in I_i$. Let $S = I_1 \cup I_2 \cup \cdots \cup I_m$. Recall that the increase due to Case 2 (in the analysis framework above) can only happen when $\tilde{n}_{\mathrm{a}} \ge 1$, i.e. only during S. Then, by Lemma 3, the increase of ψ due to Case 2 is at most $\int_{t\in S} n_{\mathrm{a}}(t) + P(s_{\mathrm{a}}(t))\,\mathrm{d}t = \sum_{i=1}^m G_{\mathrm{a}}[I_i] \le \sum_{i=1}^m \left(3 \cdot G_{\mathrm{o}}[I_i] + 2\int_{t\in I_i} k(t)\,\mathrm{d}t\right) \le 3 \cdot G_{\mathrm{o}} + 2\int_{t\in S} k(t)\,\mathrm{d}t$.

We now upper bound $\int_{t\in S} k(t)\,\mathrm{d}t$. Whenever OPT rejects a job, ψ increases by c, and then ψ stays the same until ϕ reaches ψ. Since $\phi(t)$ increases at the rate of $n_{\mathrm{a}}(t)+P(s_{\mathrm{a}}(t))$, we have $R_{\mathrm{o}} = \int_{t:\psi(t)>\phi(t)} n_{\mathrm{a}}(t) + P(s_{\mathrm{a}}(t))\,\mathrm{d}t$. Note that $k(t) = \lfloor \frac{\psi(t)}{c} \rfloor - \lfloor \frac{\phi(t)}{c} \rfloor > 0$ implies $\psi(t) > \phi(t)$. Thus, $R_{\mathrm{o}} \geq \int_{t:k(t)>0} n_{\mathrm{a}}(t) + P(s_{\mathrm{a}}(t))\,\mathrm{d}t$. At any time $t \in S$, $\tilde{n}_{\mathrm{a}}(t) > 0$ and hence $k(t) < n_{\mathrm{a}}(t)$. Then $\int_{t\in S} k(t)\,\mathrm{d}t = \int_{t:t\in S\wedge k(t)>0} k(t)\,\mathrm{d}t < \int_{t:t\in S\wedge k(t)>0} n_{\mathrm{a}}(t)\,\mathrm{d}t \leq \int_{t:k(t)>0} n_{\mathrm{a}}(t)\,\mathrm{d}t < R_{\mathrm{o}}$.

Therefore, the increase of ψ due to Case 2 is at most $3G_{\mathrm{o}} + 2R_{\mathrm{o}}$. Adding up the increase of ψ due to Case 1, i.e., R_{o}, gives $\psi(t_e) \leq 3G_{\mathrm{o}} + 3R_{\mathrm{o}}$. By Fact 2 and $\phi(t_e) \leq \psi(t_e)$, $G_{\mathrm{a}} + R_{\mathrm{a}} \leq 6(G_{\mathrm{o}} + R_{\mathrm{o}})$ and hence UPUW is 6-competitive. ◀

The rest of the section is devoted to proving Lemma 3. Before giving the potential analysis, we state a property of set $B(t)$ and show that at any time t, the size of $B(t)$ is no more than the number of active jobs in OPT by $P(T) - 1$ (Lemma 5). Without loss of generality, we assume $P(T) \geq 1$,[1] and OPT rejects a job only at its arrival time.

▶ **Property 4.** *At any time t, the set $B(t)$ only changes upon various events as follows.*

(i) *If a job j arrives and* OPT *rejects j, then $n_{\mathrm{a}} - k$ remains the same, so either B does not change or j replaces another job j' with remaining work at least $p(j)$ in B.*
(ii) *If a job j arrives and* OPT *admits j, $n_{\mathrm{a}} - k$ increases by 1, so either B remains empty, or j is added to B, or another job j' with remaining work at most $p(j)$ is added to B.*
(iii) *If* UPUW *completes a job j, $n_{\mathrm{a}} - k$ decreases by 1, so either B remains empty or j leaves B.*
(iv) *If* UPUW *rejects a job, then ϕ reaches a multiple of c, and $n_{\mathrm{a}} - k$ either remains the same or decreases by 1. Thus, either B does not change or a job in B leaves B.*

▶ **Lemma 5.** *At any time t, $\tilde{n}_{\mathrm{a}}(t) - n_{\mathrm{o}}(t) + 1 \leq P(T)$.*

We now show Lemma 5. If $\tilde{n}_{\mathrm{a}}(t) = 0$, $\tilde{n}_{\mathrm{a}}(t) = 0 \leq P(T) - 1$ (as $P(T) \geq 1$). If $s_{\mathrm{a}}(t) < T$, then $s_{\mathrm{a}}(t) = Q(n_{\mathrm{a}}(t) + 1) < T$ and hence $n_{\mathrm{a}}(t) + 1 \leq P(T)$, so $\tilde{n}_{\mathrm{a}}(t) \leq n_{\mathrm{a}}(t) \leq P(T) - 1$.

It remains to consider that $\tilde{n}_{\mathrm{a}}(t) \geq 1$ and $s_{\mathrm{a}}(t) = T$. Let t' be the last time before t such that $\tilde{n}_{\mathrm{a}}(t') = 0$ or $s_{\mathrm{a}}(t') < T$. By above, we can show that $\tilde{n}_{\mathrm{a}}(t') \leq P(T) - 1$. For any time $x \in (t', t]$, $\tilde{n}_{\mathrm{a}}(x) \geq 1$ and $s_{\mathrm{a}}(x) = T$. Let $N_o(x)$ be the set of jobs arriving during $(t', x]$ that are admitted by OPT. Suppose OPT has completed h jobs in $N_o(t)$ in $(t', t]$. Let S be the schedule obtained by running SRPT at speed T during $(t', t]$ on jobs $B(t') \cup N_o(t)$. Since SRPT maximizes the number of jobs completed by any time [14], S completes at least h jobs during $(t', t]$. As $n_{\mathrm{o}}(t) \geq |N_o(t)| - h$, the number of active jobs in S at t is at most $\tilde{n}_{\mathrm{a}}(t') + |N_o(t)| - h \leq P(T) - 1 + n_{\mathrm{o}}(t)$.

We relate the schedule of UPUW with S. At any time $x \in [t', t]$, let $B(x) = \{j_1, j_2, \cdots, j_{\tilde{n}_{\mathrm{a}}(x)}\}$, ordered in non-decreasing remaining work in UPUW; we always use job ids for tie-breaking. We can show Lemma 6 below by induction on time $[t', t]$ over various events stated in Property 4. Details will be given in the full paper. This lemma implies that at time t, the size of $B(t)$ is less than the number of active jobs in S, i.e., $\tilde{n}_{\mathrm{a}}(t) \leq P(T) - 1 + n_{\mathrm{o}}(t)$, implying Lemma 5.

▶ **Lemma 6.** *At any time $x \in (t', t]$, there is a one-to-one mapping $\rho : B(x) \to B(t') \cup N_o(x)$ such that the remaining work of each $j_i \in B(x)$ in* UPUW *is at most that of $\rho(j_i)$ in S, and $\rho(j_1), \rho(j_2), \cdots, \rho(j_{\tilde{n}_{\mathrm{a}}(x)})$ are in non-decreasing order of remaining work in S.*

[1] If $P(T) < 1$, we use the algorithm in [4] for job selection, which is 2-competitive for flow plus penalty, and always run at speed T. When the algorithm is running a job, the power is less than 1 and the number of active jobs is at least 1, so the total energy usage is at most the total flow time and hence this algorithm is 4-competitive.

We now give the potential analysis for proving Lemma 3. Recall that we are considering an interval $I = (t_1, t_2)$. Let $G_a(t)$ and $G_o(t)$ be the flow plus energy incurred from time t_1 up to time t by UPUW and OPT, respectively, for any $t \in I$. It suffices to define a potential function $\Phi(t)$ for any time $t \in I$ such that the following conditions hold: *(i) Boundary condition:* At time t_1 and t_2, $\Phi = 0$. *(ii) Discrete-event condition:* During I, when a job arrives, or a job is completed by UPUW or OPT, or a job is rejected by UPUW, $\Delta\Phi(t) \le 0$. *(iii) Running condition:* At any other time $t \in I$, $\frac{\mathrm{d}G_a(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le 3 \cdot \frac{\mathrm{d}G_o(t)}{\mathrm{d}t} + 2k(t)$. Then, Lemma 3 follows by integrating these conditions over I.

Potential function $\Phi(t)$. Consider any time t. Let $\tilde{n}_a(q,t)$ and $n_o(q,t)$ be the number of active jobs in $B(t)$ and OPT, respectively, with remaining work at least q. Note that $\tilde{n}_a(t) = \tilde{n}_a(0,t)$ and $n_o(t) = n_o(0,t)$. We will drop the parameter t from the notations when t refers to the current time clearly. Let $(\cdot)_+ = \max(\cdot, 0)$. We adapt the potential function given in [7, 3] as follows:

$$\Phi(t) = 3 \int_0^\infty \sum_{i=1}^{(\tilde{n}_a(q,t) - n_o(q,t))_+} P'(Q(i))\,\mathrm{d}q \ .$$

The boundary condition holds because at t_1 and t_2, $\tilde{n}_a = 0$, so $\tilde{n}_a(q) = 0$ for all q and $\Phi = 0$. We now check the discrete-event condition. Note that $\tilde{n}_a(x) \ge 1$ for any $x \in (t_1, t_2)$. When a job j arrives and is rejected by OPT, by Property 4(i), there are two cases: (Case 1) B does not change, then Φ does not change. (Case 2) j replaces another job j' with remaining work $q' \ge p(j)$ in B. Then $n_a(q)$ decreases by 1 for $q \in [p(j), q']$ and Φ does not increase. When a job j arrives and is admitted by OPT, by Property 4(ii), a job j' (which may be j) with remaining work $q' \le p(j)$ is added to B. Then $\tilde{n}_a(q)$ increases by 1 for $q \in [0, q'] \subseteq [0, p(j)]$ and $n_o(q)$ increases by 1 for $q \in [0, p(j)]$. Thus, $\tilde{n}_a(q) - n_o(q)$ does not increase for all q and Φ does not increase. When a job is completed by UPUW or OPT, $\tilde{n}_a(q)$ or $n_o(q)$ changes only at the single point $q = 0$, which does not affect the integration and hence Φ remains the same. Finally, when a job is rejected by UPUW, either B does not change or a job in B leaves B. For the former case, Φ does not change. For the latter case, let q' be the remaining work of the job that leaves B. Then $\tilde{n}_a(q)$ decreases by 1 for $q \in [0, q']$, and hence Φ does not increase.

It remains to check the running condition. Consider any time $t \in (t_1, t_2)$ without job arrival, completion and rejection. Let s_a and s_o be the current speeds of UPUW and OPT, respectively. To bound the rate of change of Φ, Lemma 7 below shows how Φ changes in an infinitesimal amount of time (from t to $t + \mathrm{d}t$). Its proof is based on similar arguments as in [7, 3] and will be given in the full paper.

▶ **Lemma 7.** *Consider any time t without job arrival or completion and $\tilde{n}_a \ge 1$. If $\tilde{n}_a < n_o$, then $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 0$; if $\tilde{n}_a \ge n_o$, then either* (i) $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3 \cdot P'(Q(\tilde{n}_a - n_o))(-s_a + s_o)$, *or* (ii) $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3 \cdot P'(Q(\tilde{n}_a - n_o + 1))(-s_a + s_o)$ *and $n_o \ge 1$, or* (iii) $\frac{\mathrm{d}\Phi}{\mathrm{d}t} = 0$ *and $\tilde{n}_a = n_o$.*

▶ **Lemma 8.** *At any time in (t_1, t_2) without job arrival, completion and rejection, $\frac{\mathrm{d}G_a}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3 \cdot \frac{\mathrm{d}G_o}{\mathrm{d}t} + 2k$.*

Proof. Note that during (t_1, t_2), $\tilde{n}_a \ge 1$ and $\tilde{n}_a = n_a - k$. Also, $s_a = \min(Q(n_a + 1), T)$ and hence $\frac{\mathrm{d}G_a}{\mathrm{d}t} = n_a + P(s_a) \le 2n_a + 1 \le 2n_a + \tilde{n}_a = 3\tilde{n}_a + 2k$. Similarly, $\frac{\mathrm{d}G_o}{\mathrm{d}t} = n_o + P(s_o)$. If $\tilde{n}_a < n_o$, by Lemma 7, $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 0$ and thus $\frac{\mathrm{d}G_a}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3\tilde{n}_a + 2k < 3n_o + 2k \le 3\frac{\mathrm{d}G_o}{\mathrm{d}t} + 2k$. Otherwise, if $\tilde{n}_a \ge n_o$, we consider the three cases in Lemma 7, where we need the upper bound on $\tilde{n}_a - n_o$ (Lemma 5).

Case (i): $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3P'(Q(\tilde{n}_a - n_o))(-s_a + s_o)$. By a lemma given in [7] (stated as Lemma 9 below), $\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 3(-s_a + Q(\tilde{n}_a - n_o))P'(Q(\tilde{n}_a - n_o)) + 3P(s_o) - 3(\tilde{n}_a - n_o)$. If $s_a = T$, by Lemma 5,

$s_a = T \geq Q(\tilde{n}_a - n_o + 1) \geq Q(\tilde{n}_a - n_o)$; otherwise, $s_a = Q(n_a + 1) \geq Q(\tilde{n}_a + 1) \geq Q(\tilde{n}_a - n_o)$. Thus, $\frac{d\Phi}{dt} \leq 3(n_o + P(s_o)) - 3\tilde{n}_a$ and hence $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \leq 3\tilde{n}_a + 2k + \frac{d\Phi}{dt} \leq 3(n_o + P(s_o)) + 2k = 3\frac{dG_o}{dt} + 2k$.

Case (ii): $\frac{d\Phi}{dt} \leq 3P'(Q(n_a - n_o + 1))(-s_a + s_o)$ and $n_o \geq 1$. By Lemma 9, $\frac{d\Phi}{dt} \leq 3(-s_a + Q(\tilde{n}_a - n_o + 1))P'(Q(\tilde{n}_a - n_o + 1)) + 3P(s_o) - 3(\tilde{n}_a - n_o + 1)$. If $s_a = T$, by Lemma 5, $s_a = T \geq Q(\tilde{n}_a - n_o + 1)$; otherwise, $s_a = Q(n_a + 1) \geq Q(\tilde{n}_a + 1) \geq Q(\tilde{n}_a - n_o + 1)$. Thus, $\frac{d\Phi}{dt} \leq 3(n_o + P(s_o)) - 3\tilde{n}_a - 3$, and hence $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \leq 3\tilde{n}_a + 2k + \frac{d\Phi}{dt} \leq 3(n_o + P(s_o)) + 2k - 3 \leq 3\frac{dG_o}{dt} + 2k$.

Case (iii): $\frac{d\Phi}{dt} = 0$ and $\tilde{n}_a = n_o$. Then $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \leq 3\tilde{n}_a + 2k = 3n_o + 2k \leq 3\frac{dG_o}{dt} + 2k$. ◀

Below is the lemma given in [7], which is used in the proof of Lemma 8.

▶ **Lemma 9.** [7] *Let P be a strictly increasing, strictly convex, continuous and differentiable function. Let $i, s_a, s_o \geq 0$ be any real. Then, $P'(Q(i))(-s_a + s_o) \leq (-s_a + Q(i))P'(Q(i)) + P(s_o) - i$.*

3 Arbitrary Penalty and Arbitrary Weight

This section considers jobs of arbitrary penalty and arbitrary weight in the following two models. In the *fixed-speed* model, the processor always runs at speed 1 and energy is not a concern. The objective is to minimize the user cost, i.e., total weighted flow plus penalty. In the *speed scaling* model, the processor can scale its speed with an arbitrary power function $P(s)$ and maximum speed T. Then the objective is to minimize the user cost plus energy.

In the speed scaling model, we give an $O((1+\frac{1}{\epsilon})^2)$-competitive algorithm for weighted flow plus penalty plus energy, using $(1+\epsilon)^2$-speedup processor for any $\epsilon > 0$. In the fixed-speed model, we give a $(1+\epsilon)^2$-speed $O((1+\frac{1}{\epsilon})^2)$-competitive algorithm for weighted flow plus penalty. This improves the $(1+\epsilon)$-speed $O(\frac{1}{\epsilon}(\log W + \log C)^2)$-competitive result in [4].

Fractional weighted flow. To obtain these results, we will first focus on the objective of total *fractional* weighted flow, and then convert the result for (integral) weighted flow. At any time t, the fractional weight of an active job j, denoted by $w(j,t)$, is its weight times its remaining fraction, i.e., $w(j,t) = w(j) \cdot \frac{q(j,t)}{p(j)}$, where $q(j,t)$ is the remaining size of j at t. Then the fractional weighted flow of job j is $\int_{r(j)}^{\infty} w(j,t)\,dt$, and hence the total fractional weighted flow is $\int_0^{\infty} w_a(t)\,dt$, where $w_a(t)$ is the total fractional weight of active jobs at time t.

HDF and future cost. Under a fixed speed function, HDF (highest density first) minimizes fractional weighted flow [8]. Our algorithm will always rejects a job at its arrival time and processes the admitted jobs using HDF. Furthermore, at any time, the processor always scales its speed according to the total fractional weight w of the active jobs, and we denote this speed by $s(w)$ (for fixed-speed processor, $s(w)$ is a constant). Consider any time t. Let $w_a(q,t)$ be the total fractional weight of active jobs with inverse density at least q. Then we can define a *future cost* $\widehat{\Phi}_a(t)$ to capture the total fractional weighted flow to serve the current active jobs if no jobs arrive in the future [12]:

$$\widehat{\Phi}_a(t) = \int_{q=0}^{\infty} \int_{x=0}^{w_a(q,t)} \frac{x}{s(x)}\,dx\,dq\ .$$

Algorithm HDF-AC. We focus on the objective of fractional weighted flow and define the algorithm HDF-AC that works for both the speed scaling and fixed-speed models. Let $\epsilon > 0$ be a constant. Consider any time t.

Job execution: Let $w_a(t)$ and $s_a(t)$ be the total fractional weight of active jobs and the speed of HDF-AC. In the fixed-speed model, we use $(1+\epsilon)$-speed processor, so $s_a(t) = 1 + \epsilon$;

in the speed scaling model, we use $(1+\epsilon)$-speedup processor and set $s_{\mathrm{a}}(t) = (1+\epsilon)\cdot \min(Q(w_{\mathrm{a}}(t)), T)$. Then, HDF-AC runs the admitted jobs using HDF at speed $s_{\mathrm{a}}(t)$.

Admission control: Let $w_{\mathrm{a}}(q,t)$ be the total fractional weight of active jobs with inverse density at least q. Let $f(x) = \frac{x}{\min(Q(x),T)}$ in the speed scaling model, and $f(x) = x$ in the fixed-speed model. Then the future cost at time t is

$$\widehat{\Phi}_a(t) = \frac{1}{1+\epsilon}\cdot \int_{q=0}^{\infty}\int_{x=0}^{w_{\mathrm{a}}(q,t)} f(x)\,\mathrm{d}x\,\mathrm{d}q\ .$$

When a job j arrives, let $\Delta\widehat{\Phi}_a(t)$ be the increase in $\widehat{\Phi}_a(t)$ if j is admitted. More precisely, let $d(j) = p(j)/w(j)$ be the inverse density of j. Then $\Delta\widehat{\Phi}_a(t) = \frac{1}{1+\epsilon}\cdot\int_{q=0}^{d(j)}\int_{x=w_{\mathrm{a}}(q,t)}^{w_{\mathrm{a}}(q,t)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q$. HDF-AC discards j if $v(j) \le \Delta\widehat{\Phi}_a(t)$; otherwise, j is admitted.

Our main result is the following theorem.

▶ **Theorem 10.** *Consider any $\epsilon > 0$.* (i) *In the speed scaling model, HDF-AC is $(8+\frac{12}{\epsilon})$-competitive for fractional weighted flow plus penalty plus energy, when using $(1+\epsilon)$-speedup processor.* (ii) *In the fixed-speed model, HDF-AC is $(3+\frac{6}{\epsilon})$-competitive for fractional weighted flow plus penalty, when using $(1+\epsilon)$-speed processor.*

Though the objectives for Theorem 10 (i) and (ii) are different, we present an analysis framework that works for both objectives. Let OPT be the optimal offline schedule for the corresponding objective. Without loss of generality, we can assume that OPT rejects a job at its arrival. Let $w_{\mathrm{o}}(t)$ and $s_{\mathrm{o}}(t)$ be the total fractional weight of active jobs and the speed of OPT. In the speed scaling model, the objective is fractional weighted flow plus penalty plus energy. We further assume that OPT runs the BCP algorithm [7], i.e., at any time t, OPT runs the admitted jobs using HDF at speed $s_{\mathrm{o}}(t) = \min(Q(w_{\mathrm{o}}(t)), T)$. Since BCP is 2-competitive for fractional weighted flow plus energy [7], such assumption on OPT only increases the competitive ratio by a factor of 2. In the fixed-speed model, the objective is fractional weighted flow plus penalty. We further assume that OPT runs HDF at speed $s_{\mathrm{o}}(t) = 1$, since HDF minimizes fractional weighted flow [8].

Since OPT runs HDF, we can define its future cost similarly. At any time t, let $w_{\mathrm{o}}(q,t)$ be the total fractional weight of active jobs with inverse density at least q. Recall that $f(x) = \frac{x}{\min(Q(x),T)}$ in the speed scaling model, and $f(x) = x$ in the fixed-speed model. Then the future cost of OPT at time t is

$$\widehat{\Phi}_o(t) = \int_{q=0}^{\infty}\int_{x=0}^{w_{\mathrm{o}}(q,t)} f(x)\,\mathrm{d}x\,\mathrm{d}q\ .$$

Overview of analysis. Our analysis exploits amortization and potential functions. We split the objective into two parts; R denotes the penalty and G denotes the fractional weighted flow (plus energy). Let $G_{\mathrm{a}}(t)$ and $G_{\mathrm{o}}(t)$ denote the objective G incurred up to time t by HDF-AC and OPT, respectively. Define $R_{\mathrm{a}}(t)$ and $R_{\mathrm{o}}(t)$ similarly for the penalty R. To show that HDF-AC is (c_1+c_2)-competitive for the objective $G+R$ against OPT, 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) Completion condition:* When a job is completed by HDF-AC or OPT, $\Delta\Phi(t) \le 0$. *(iii) Arrival condition:* When a job arrives, $\Delta R_{\mathrm{a}}(t) + \Delta\Phi(t) \le c_1\cdot(\Delta\widehat{\Phi}_o(t) + \Delta R_{\mathrm{o}}(t))$, where $\Delta\widehat{\Phi}_o(t)$ is the change in the future cost of OPT at time t. *(iv) Running condition:* At any other time, $\frac{\mathrm{d}G_{\mathrm{a}}(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le c_2\cdot\frac{\mathrm{d}G_{\mathrm{o}}(t)}{\mathrm{d}t}$.

To see the correctness, note that $R_{\mathrm{a}}(t)$ and $R_{\mathrm{o}}(t)$ changes discretely only at job arrivals, and $G_{\mathrm{a}}(t)$ and $G_{\mathrm{o}}(t)$ changes continuously at other times. Let t_e be the time when all jobs are completed by both HDF-AC and OPT. Since the future cost $\widehat{\Phi}_o(t)$ captures the fractional

weighted flow incurred by OPT to serve the active jobs at t, we have $\int_0^{t_e} \Delta\widehat{\Phi}_o(t)\,\mathrm{d}t \le G_{\mathrm{o}}(t_e)$. Therefore, the correctness follows from integrating these conditions over time, which gives

$$G_{\mathrm{a}}(t_e) + R_{\mathrm{a}}(t_e) \le c_1 \cdot (\int_0^{t_e} \Delta\widehat{\Phi}_o(t)\,\mathrm{d}t + R_{\mathrm{o}}(t_e)) + c_2 \cdot G_{\mathrm{o}}(t_e) \le (c_1+c_2)\cdot(G_{\mathrm{o}}(t_e) + R_{\mathrm{o}}(t_e))\ .$$

Potential function. We now define a general form of $\Phi(t)$ that works for both objectives. Consider any time t. Recall that $f(x) = \frac{x}{\min(Q(x),T)}$ in the speed scaling model, and $f(x) = x$ in the fixed-speed model. The potential function Φ is defined as

$$\Phi(t) = \frac{2}{\epsilon} \cdot \int_{q=0}^{\infty} \int_{x=0}^{(w_{\mathrm{a}}(q,t)-w_{\mathrm{o}}(q,t))_+} f(x)\,\mathrm{d}x\,\mathrm{d}q\ .$$

The boundary and completion conditions hold obviously. We now check the arrival condition. We drop the parameter t from all notations when it is clear that t refers to the current time.

▶ **Lemma 11.** *When a job j arrives, $\Delta R_{\mathrm{a}} + \Delta\Phi \le (2+\frac{4}{\epsilon})\cdot(\Delta\widehat{\Phi}_o + \Delta R_{\mathrm{o}})$.*

Proof. Let $d(j) = p(j)/w(j)$ be the inverse density of job j. If HDF-AC admits this job, then $w_{\mathrm{a}}(q)$ increases by $w(j)$ for $q \in [0, d(j)]$. Similarly, if OPT admits this job, then $w_{\mathrm{o}}(q)$ increases by $w(j)$ for $q \in [0, d(j)]$. Now, we consider the following two cases.

Case 1: OPT admits j. In this case, $\Delta R_{\mathrm{o}} = 0$. If HDF-AC also admits j, then $w_{\mathrm{a}}(q) - w_{\mathrm{o}}(q)$ remains the same for all q, so $\Delta\Phi = 0$. Since $\Delta\widehat{\Phi}_o \ge 0$, $\Delta R_{\mathrm{a}} + \Delta\Phi = 0 \le (2+\frac{4}{\epsilon})\cdot(\Delta\widehat{\Phi}_o + \Delta R_{\mathrm{o}})$.

Otherwise, HDF-AC rejects j. We analyze using techniques in [12]. Note that $\Delta\widehat{\Phi}_o = \int_{q=0}^{d(j)} \int_{x=w_{\mathrm{o}}(q)}^{w_{\mathrm{o}}(q)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q$. The change of Φ due to OPT is

$$-\frac{2}{\epsilon} \cdot \int_{q=0}^{d(j)} \int_{x=(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q)-w(j))_+}^{(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q))_+} f(x)\,\mathrm{d}x\,\mathrm{d}q\ .$$

Note that $\Delta R_{\mathrm{a}} = v(j) \le \frac{1}{1+\epsilon}\int_{q=0}^{d(j)} \int_{x=w_{\mathrm{a}}(q)}^{w_{\mathrm{a}}(q)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q \le \frac{2}{\epsilon}\int_{q=0}^{d(j)} \int_{x=w_{\mathrm{a}}(q)}^{w_{\mathrm{a}}(q)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q$, and the change of Φ due to HDF-AC is zero. Thus,

$$\Delta R_{\mathrm{a}} + \Delta\Phi \le \frac{2}{\epsilon} \cdot \int_{q=0}^{d(j)} \left(\int_{x=w_{\mathrm{a}}(q)}^{w_{\mathrm{a}}(q)+w(j)} f(x)\,\mathrm{d}x - \int_{x=(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q)-w(j))_+}^{(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q))_+} f(x)\,\mathrm{d}x \right) \mathrm{d}q\ .$$

It was shown in [12] that if the function f satisfies that $f(0) \ge 0$ and f is increasing and subadditive, i.e., for any $a, b \ge 0$, $f(a+b) \le f(a) + f(b)$, then $\int_{x=w_{\mathrm{a}}(q)}^{w_{\mathrm{a}}(q)+w(j)} f(x)\,\mathrm{d}x - \int_{x=(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q)-w(j))_+}^{(w_{\mathrm{a}}(q)-w_{\mathrm{o}}(q))_+} f(x)\,\mathrm{d}x \le 2\int_{x=w_{\mathrm{o}}(q)}^{w_{\mathrm{o}}(q)+w(j)} f(x)\,\mathrm{d}x$. In the fixed-speed model, $f(x) = x$ obviously satisfies these conditions. In the speed scaling model, $f(x) = \frac{x}{\min(Q(x),T)}$. It was also shown in [12] that $\frac{x}{Q(x)}$ is increasing and subadditive. Clearly, $f(x)$ is also increasing. Consider any $a, b \ge 0$. If $a + b \le P(T)$, it follows directly that $f(a+b) \le f(a) + f(b)$; otherwise, $f(a+b) = \frac{a+b}{T} \le \frac{a}{\min(Q(a),T)} + \frac{b}{\min(Q(b),T)} = f(a) + f(b)$. Therefore, in both cases, we can apply the inequality to get that $\Delta R_{\mathrm{a}} + \Delta\Phi \le \frac{4}{\epsilon} \cdot \int_{q=0}^{d(j)} \int_{x=w_{\mathrm{o}}(q)}^{w_{\mathrm{o}}(q)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q = \frac{4}{\epsilon}\cdot\Delta\widehat{\Phi}_o \le (2+\frac{4}{\epsilon})\cdot(\Delta\widehat{\Phi}_o + \Delta R_{\mathrm{o}})$.

Case 2: OPT rejects j. In this case, $\Delta R_{\mathrm{o}} = v(j)$, $\Delta\widehat{\Phi}_o = 0$, and the change of Φ due to OPT is zero. Similarly, if HDF-AC admits j, then $\Delta R_{\mathrm{a}} = 0$ and the change of Φ due to HDF-AC is

$$\frac{2}{\epsilon} \cdot \int_{q=0}^{d(j)} \int_{x=w_{\mathrm{a}}(q)}^{w_{\mathrm{a}}(q)+w(j)} f(x)\,\mathrm{d}x\,\mathrm{d}q\ ,$$

which is exactly $2(1+\frac{1}{\epsilon})$ times the increase of $\widehat{\Phi}_a$ and is therefore at most $2(1+\frac{1}{\epsilon})v(j)$. Otherwise, if HDF-AC also rejects j, $\Delta R_{\mathrm{a}} = v(j)$ and the change of Φ due to HDF-AC is zero. In both cases, $\Delta R_{\mathrm{a}} + \Delta\Phi \le (2+\frac{2}{\epsilon})\cdot v(j) \le (2+\frac{4}{\epsilon})\cdot(\Delta\widehat{\Phi}_o + \Delta R_{\mathrm{o}})$. ◀

It remains to show the running condition. Consider any time t without job arrival or completion. Let s_a and s_o be the current speeds of HDF-AC and OPT, respectively. To bound the rate of change of Φ, Lemma 12 below shows how Φ changes in an infinitesimal amount of time (from t to $t + dt$). Its proof is based on similar arguments as in [7, 12] and will be given in the full paper.

▶ **Lemma 12.** *Consider any time without job arrival or completion.* (i) *If* $w_a < w_o$, *then* $\frac{d\Phi}{dt} \le 0$. (ii) *If* $w_a > w_o$, *then* $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot f(w_a - w_o) \cdot (-s_a + s_o)$. (iii) *If* $w_a = w_o$, *then* $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot f(w_o) \cdot s_o$.

We are ready to show the running condition for the speed scaling model (Lemma 13) and for the fixed-speed model (Lemma 14).

▶ **Lemma 13.** *In the speed scaling model, at any time without job arrival or completion,* $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \le (2 + \frac{2}{\epsilon}) \cdot \frac{dG_o}{dt}$.

Proof. Since $s_a = (1+\epsilon)\min(Q(w_a), T) \le (1+\epsilon)Q(w_a)$ and HDF-AC is using a $(1+\epsilon)$-speedup processor, $P(s_a) \le w_a$ and $\frac{dG_a}{dt} \le 2w_a$. By the assumption of OPT, $s_o = \min(Q(w_o), T)$ and $\frac{dG_o}{dt} \ge w_o$. We now consider the three cases stated in Lemma 12. Recall that $f(x) = \frac{x}{\min(Q(x),T)}$.
Case (i): $w_a < w_o$. By Lemma 12, $\frac{d\Phi}{dt} \le 0$, so $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \le 2w_a < 2w_o \le (2 + \frac{2}{\epsilon}) \cdot \frac{dG_o}{dt}$.
Case (ii): $w_a > w_o$. Note that Q is increasing. By Lemma 12, $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot \frac{w_a - w_o}{\min(Q(w_a - w_o), T)}(-(1+\epsilon)\min(Q(w_a), T) + \min(Q(w_o), T)) \le -\frac{2}{\epsilon} \cdot (w_a - w_o)\frac{\epsilon \cdot \min(Q(w_a),T)}{\min(Q(w_a - w_o),T)} \le 2w_o - 2w_a$. Thus, $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \le 2w_a + 2w_o - 2w_a \le (2 + \frac{2}{\epsilon}) \cdot \frac{dG_o}{dt}$.
Case (iii): $w_a = w_o$. By Lemma 12, $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot \frac{w_o}{\min(Q(w_o),T)} \cdot \min(Q(w_o), T) = \frac{2}{\epsilon} \cdot w_o$. Thus, $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \le 2w_a + \frac{2}{\epsilon} \cdot w_o = (2 + \frac{2}{\epsilon}) \cdot w_o \le (2 + \frac{2}{\epsilon}) \cdot \frac{dG_o}{dt}$. ◀

▶ **Lemma 14.** *In the fixed-speed model, at any time without job arrival or completion,* $\frac{dG_a}{dt} + \frac{d\Phi}{dt} \le (1 + \frac{2}{\epsilon}) \cdot \frac{dG_o}{dt}$.

Proof. It suffices to show that $w_a + \frac{d\Phi}{dt} \le (1 + \frac{2}{\epsilon}) \cdot w_o$. Recall that $s_a = 1 + \epsilon$, $s_o = 1$ and $f(x) = x$. We now consider the three cases stated in Lemma 12.
Case (i): $w_a < w_o$. By Lemma 12, $\frac{d\Phi}{dt} \le 0$, so $w_a + \frac{d\Phi}{dt} \le w_a \le w_o \le (1 + \frac{2}{\epsilon}) \cdot w_o$.
Case (ii): $w_a > w_o$. By Lemma 12, $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot (w_a - w_o) \cdot (-(1+\epsilon) + 1) = 2w_o - 2w_a$. Therefore, $w_a + \frac{d\Phi}{dt} \le w_a + 2w_o - 2w_a \le w_o \le (1 + \frac{2}{\epsilon}) \cdot w_o$.
Case (iii): $w_a = w_o$. By Lemma 12, $\frac{d\Phi}{dt} \le \frac{2}{\epsilon} \cdot w_o$. Thus, $w_a + \frac{d\Phi}{dt} \le w_a + \frac{2}{\epsilon} \cdot w_o = (1 + \frac{2}{\epsilon}) \cdot w_o$. ◀

In the speed scaling model, by Lemmas 11 and 13, HDF-AC is $(4 + \frac{6}{\epsilon})$-competitive against OPT. Recall that OPT uses BCP and thus is 2-approximate. Therefore, Theorem 10 (i) follows. In the fixed-speed model, by Lemmas 11 and 14, HDF-AC is $(3 + \frac{6}{\epsilon})$-competitive against OPT, which is the actual optimal schedule. Thus, Theorem 10 (ii) follows.

Online algorithm for integral weighted flow. We now convert the result of Theorem 10 for the objective of (integral) weighted flow. Since HDF is $(1+\epsilon)$-speed $(1 + \frac{1}{\epsilon})$-competitive for weighted flow on a fixed speed processor [8] and the fractional weighted flow of any schedule (including OPT) is always at most its (integral) weighted flow, we use the following online algorithm HDF-AC*: HDF-AC* keeps a simulated copy of HDF-AC on the same job instance. It always follows the admission control of HDF-AC. At any time, HDF-AC* runs at speed $(1+\epsilon)$ faster than the simulated HDF-AC, but selects the job to run using HDF on its own active jobs.

The following performance guarantee of HDF-AC* follows directly from Theorem 10.

▶ **Corollary 15.** *Consider any $\epsilon > 0$.* (i) *In the speed scaling model, HDF-AC* is $(1+\frac{1}{\epsilon})(8+\frac{12}{\epsilon})$-competitive for weighted flow plus penalty plus energy, when using $(1+\epsilon)^2$-speedup processor.* (ii) *In the fixed-speed model, HDF-AC* is $(1+\frac{1}{\epsilon})(3+\frac{6}{\epsilon})$-competitive for weighted flow plus penalty, when using $(1+\epsilon)^2$-speed processor.*

4 Lower Bound for Arbitrary Penalty Jobs

This section gives the lower bound result. Assuming $P(s) = s^\alpha$, we show that the competitive ratio of any algorithm must grow with α, i.e., the steepness of the power function. This implies that no $O(1)$-competitive algorithm exists for arbitrary power function.

▶ **Theorem 16.** *Consider minimizing flow plus energy plus penalty. For power function $P(s) = s^\alpha$, if T is unbounded, any algorithm is $\Omega(\alpha^{1/2-\epsilon})$-competitive for any $0 < \epsilon < \frac{1}{2}$.*

Proof. Let A be any algorithm and OFF be the offline adversary. Let $k \geq 1$ be some constant depending on α (to be defined later). At time 0, the adversary releases two streams of jobs, namely Stream 1 and Stream 2. Stream 1 contains k^2 jobs of size 1 and penalty k^2, each released at time i, where $0 \leq i \leq k^2 - 1$. Stream 2 contains k job of size k and penalty k^5, each released at time jk, where $0 \leq j \leq k-1$. The penalty of Stream 2 jobs is large enough such that A is not competitive if any one of them is rejected. Therefore, A runs Stream 2 jobs one by one (in SRPT) in their arrival order. Depending on the number of Stream 2 jobs remaining in A at time k^2, the adversary may release Stream 3, which contains $\frac{k^4}{\delta}$ job of size $\delta = \frac{1}{k}$ and penalty k^5, each released at time $k^2 + i\delta$, where $0 \leq i \leq \frac{k^4}{\delta} - 1$.

Case 1: At time k^2, A has less than $\frac{k}{2}$ Stream 2 jobs remaining. In this case, the adversary does not release Stream 3. OFF can always run at speed 1 and completes the Stream 1 jobs one by one in $[0, k^2]$ and then completes the Stream 2 jobs one by one in $[k^2, 2k^2]$. Thus, the total flow of OFF is at most $k^2 \cdot 1 + k \cdot (k^2 + k) = O(k^3)$. Since OFF always consume power $1^\alpha = 1$, which is at most the number of active jobs at that time, the energy usage of OFF is at most its flow. As OFF does not reject any job, the flow plus energy plus penalty of OFF is $O(k^3)$.

Consider the schedule of A. If A rejects at least one Stream 2 jobs, the penalty of A is at least k^5. If A rejects more than $\frac{k^2}{4}$ Stream 1 jobs, the penalty of A is at least $\frac{k^2}{4} \cdot k^2 = \Omega(k^4)$. If A has at least $\frac{k^2}{8}$ Stream 1 jobs remaining at time k^2, the flow of these jobs is at least $\sum_{i=1}^{k^2/8} i = \Omega(k^4)$. In all of the above three cases, the competitive ratio of A is $\Omega(k)$. Otherwise, A does not reject any Stream 2 job, and A rejects at most $\frac{k^2}{4}$ Stream 1 jobs, and there are less than $\frac{k^2}{8}$ Stream 1 jobs remaining at time k^2. Thus, during $[0, k^2]$, A has completed at least $k^2 - \frac{k^2}{4} - \frac{k^2}{8} = \frac{5k^2}{8}$ Stream 1 jobs and at least $k - \frac{k}{2} = \frac{k}{2}$ Stream 2 jobs. The work done of A during $[0, k^2]$ is at least $\frac{5k^2}{8} + \frac{k}{2} \cdot k = \frac{9k^2}{8}$. By the convexity of the power function s^α, running at a fixed speed minimizes the energy usage and thus the energy usage of A is at least $(\frac{9k^2}{8}/k^2)^\alpha \cdot k^2 = (\frac{9}{8})^\alpha k^2$. Thus, the competitive ratio of A is $\Omega((\frac{9}{8})^\alpha \cdot \frac{1}{k})$.

Case 2: At time k^2, A has at least $\frac{k}{2}$ Stream 2 jobs remaining. In this case, the adversary releases Stream 3. Similar to Stream 2, without loss of generality, A works on Stream 3 jobs one by one (in SRPT). OFF can reject all Stream 1 jobs and then always run at speed 1 to complete the Stream 2 jobs one by one in $[0, k^2]$ and then completes the Stream 3 jobs one by one in $[k^2, k^2 + k^4]$. Thus, the total penalty of OFF is $k^2 \cdot k^2 = k^4$ and total flow of OFF is at most $k \cdot k + \frac{k^4}{\delta} \cdot \delta = k^2 + k^4 = O(k^4)$. Since OFF always consume power $1^\alpha = 1$, which is at most the number of active jobs at that time, the energy usage of OFF is at most its flow. Therefore, the flow plus energy plus penalty of OFF is $O(k^4)$.

Consider the schedule of A. If A rejects at least one Stream 2 or Stream 3 job, the penalty of A is at least k^5. If at time k^2+k^4, A has at least $\frac{k}{4}$ Stream 2 jobs remaining, the flow of these jobs is at least $\frac{k}{4}\cdot k^4 = \Omega(k^5)$. If at time k^2+k^4, A has at least $\frac{k^2}{8\delta}$ Stream 3 jobs remaining, the flow of these jobs is at least $\delta\cdot\sum_{i=1}^{k^2/8\delta} i = \Omega(\frac{k^4}{\delta}) = \Omega(k^5)$. In all of the above three cases, the competitive ratio of A is $\Omega(k)$. Otherwise, A does not reject any Stream 2 and Stream 3 job, and at time k^2+k^4, there are less than $\frac{k}{4}$ Stream 2 jobs and less than $\frac{k^2}{8\delta}$ Stream 3 jobs remaining. Thus, A has completed more than $\frac{k}{2}-\frac{k}{4}=\frac{k}{4}$ Stream 2 jobs and more than $\frac{k^4}{\delta}-\frac{k^2}{8\delta}$ Stream 3 jobs during $[k^2, k^2+k^4]$. Since A runs Stream 2 jobs and Stream 3 jobs by SRPT, respectively, the total work done during $[k^2, k^2+k^4]$ is at least $\frac{k}{4}\cdot k + (\frac{k^4}{\delta}-\frac{k^2}{8\delta})\cdot\delta = k^4+\frac{k^2}{8}$. Since running at a fixed speed minimizes the energy usage, the energy usage of A is at least $k^4\cdot((k^4+\frac{k^2}{8})/k^4)^\alpha = \Omega(k^4\cdot(1+\frac{1}{8k^2})^\alpha)$ and hence A is $\Omega((1+\frac{1}{8k^2})^\alpha)$-competitive.

Therefore, A is $\Omega(\min(k, (\frac{9}{8})^\alpha(\frac{1}{k}), (1+\frac{1}{8k^2})^\alpha))$-competitive. We set $k=\alpha^{\frac{1}{2}-\epsilon}$ for $0<\epsilon<\frac{1}{2}$. Since $(1+\frac{1}{8y})^y$ is increasing with y, the competitive ratio of A is $\Omega(\min(\alpha^{\frac{1}{2}-\epsilon}, (\frac{9}{8})^\alpha/\alpha^{\frac{1}{2}-\epsilon}, (1+\frac{1}{8\alpha^{1-2\epsilon}})^{\alpha^{1-2\epsilon}\cdot\alpha^{2\epsilon}})) = \Omega(\min(\alpha^{\frac{1}{2}-\epsilon}, (\frac{9}{8})^\alpha/\alpha^{\frac{1}{2}-\epsilon}, (1+\frac{1}{8})^{\alpha^{2\epsilon}})) = \Omega(\alpha^{\frac{1}{2}-\epsilon})$. ◀

References

1. S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.
2. S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, 3(4):49, 2007.
3. L. Andrew, A. Wierman, and A. Tang. Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):39–41, 2009.
4. N. Bansal, A. Blum, S. Chawla, and K. Dhamdhere. Scheduling for flow-time with admission control. In *Proc. ESA*, pages 43–54, 2003.
5. D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J. D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.
6. N. Bansal, H. L. Chan, T. W. Lam, and L. K. Lee. Scheduling for speed bounded processors. In *Proc. ICALP*, pages 409–420, 2008.
7. N. Bansal, H. L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. SODA*, pages 693–701, 2009.
8. 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.
9. N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. *SIAM Journal on Computing*, 39(4):1294–1308, 2009.
10. H. L. Chan, J. Edmonds, T. W. Lam, L. K. Lee, A. Marchetti-Spaccamela, and K. Pruhs. Nonclairvoyant speed scaling for flow and energy. In *Proc. STACS*, pages 255–264, 2009.
11. S. H. Chan, T. W. Lam, and L. K. Lee. Non-clairvoyant speed scaling for weighted flow time. In *Proc. ESA*, pages 23–35, 2010.
12. A. Gupta, R. Krishnaswamy, and K. Pruhs. Scalably scheduling power-heterogeneous processors. In *Proc. ICALP*, 312–323, 2010.
13. T. W. Lam, L. K. Lee, I. To, and P. Wong. Speed scaling functions for flow time scheduling based on active job count. In *Proc. ESA*, pages 647–659, 2008.
14. L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.
15. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. FOCS*, pages 374–382, 1995.