Non-clairvoyant Weighted Flow Time Scheduling with Rejection Penalty

Ho-Leung Chan[*]
University of Hong Kong,
Hong Kong
hlchan@cs.hku.hk

Sze-Hang Chan
University of Hong Kong,
Hong Kong
shchan@cs.hku.hk

Tak-Wah Lam[†]
University of Hong Kong,
Hong Kong
twlam@cs.hku.hk

Lap-Kei Lee[‡]
University of Hong Kong,
Hong Kong
lklee@cs.hku.hk

Jianqiao Zhu
University of Hong Kong,
Hong Kong
jqzhu@cs.hku.hk

ABSTRACT

This paper initiates the study of online scheduling with rejection penalty in the non-clairvoyant setting, i.e., the size (processing time) of a job is not assumed to be known at its release time. In the rejection penalty model, jobs can be rejected with a penalty, and 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. Previous work on minimizing the total user cost focused on the clairvoyant single-processor setting [2, 8] and has produced $O(1)$-competitive online algorithm for jobs with arbitrary weights and penalties. This paper gives the first non-clairvoyant algorithms that are $O(1)$-competitive for minimizing the total user cost on a single processor and multi-processors, when using slightly faster (i.e., $(1+\epsilon)$-speed for any $\epsilon > 0$) processors. Note that if no extra speed is allowed, no online algorithm can be $O(1)$-competitive even for minimizing (unweighted) flow time alone. The new user cost results can also be regarded as a generalization of previous non-clairvoyant results on minimizing weighted flow time alone (WSETF [4] for a single processor; WLAPS [14] for multi-processors).

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 an arbitrary increasing function of speed. A scheduling algorithm has to decide job rejection and determine the order and speed of job execution. It is interesting to study the tradeoff between the above-mentioned user cost and energy. This paper gives two O(1)-competitive non-clairvoyant algorithms for minimizing the user cost plus energy on a single processor and multi-processors, respectively.

[*]The research of Ho-Leung Chan was partially supported by GRF Grant HKU710210E.

[†]The research of Tak-Wah Lam was partially supported by HKU Grant 201109176197.

[‡]Part of the work was done when Lap-Kei Lee was working in MADALGO (Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation), Aarhus University, Denmark.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance Attributes; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

Keywords

Online scheduling, competitive analysis, weighted flow time, non-clairvoyant scheduling, rejection penalty

1. INTRODUCTION

It is common that servers prioritize their jobs and reject some low-priority jobs during peak load to meet the performance guarantee. Serving too many jobs prolongs their individual response time, yet rejecting jobs would cause users' inconvenience and waste the processing power already spent on them. To study the tradeoff between response time and rejection penalty, Bansal et al. [2] and Chan et al. [8] considered flow-time scheduling on a single processor when jobs can be rejected at some penalty. Jobs arrive online with arbitrary sizes, weights and penalties, and a scheduler 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 (or simply flow) of a job is the time elapsed since the job is released until it is completed or rejected. The objective is to minimize the total user cost of all jobs.

Bansal et al. [2] showed that any online algorithm is $\Omega(\max\{n^{\frac{1}{4}}, C^{\frac{1}{2}}\})$-competitive, where n is the number of jobs and C is the max-min ratio of job penalties. In view of the lower bound, they consider giving the online algorithm a slightly faster processor. Using a $(1+\epsilon)$-speed processor for any $\epsilon > 0$, they gave an $O(\frac{1}{\epsilon}(\log W + \log C)^2)$-competitive algorithm where W is the max-min ratio of job weights. Recently, Chan et al. [8] improved the competitive ratio to $O((1+\frac{1}{\epsilon})^2)$ when using a $(1+\epsilon)^2$-speed processor, which is independent of W and C. These two results are based on job rejection policies that know the size of a job at its re-

	Weighted flow	User cost	Weighted flow + energy	User cost + energy
Single processor	$1+\frac{1}{\epsilon}$ [4]	$12(1+\frac{1}{\epsilon})^2$ [†]	$8(1+\frac{1}{\epsilon})^2$ [7]	$36(1+\frac{1}{\epsilon})^2$ [†]
$m > 1$ processors	$8(1+\frac{1}{\epsilon})^2$ [14]	$20(1+\frac{1}{\epsilon})^2$ [†]	$12(\log m + 2)(1+\frac{1}{\epsilon})^2$ [†] (using $2(1+\epsilon)$-speed processors)	

Table 1: Competitive ratios of non-clairvoyant scheduling for different objectives involving weighted flow time. Recall that user cost equals weighted flow plus penalty. All results assume using a faster processor and, unless specified otherwise, are using $(1+\epsilon)$-speed processors. Our new results are marked with [†].

lease time and its remaining size at any time, i.e., they only work in the clairvoyant setting. In this paper, we extend the study of rejection penalty to the *non-clairvoyant* setting, where the size of a released job is not known until the job is completed. Such a setting is natural from the viewpoint of operating systems.

Non-clairvoyant flow-time scheduling on a single processor. 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 the non-clairvoyant setting, even for unweighted jobs, any algorithm is $\Omega(n^{1/3})$-competitive [12] for minimizing the total flow, where n is the number of jobs. Using a slightly faster processor, Kalyanasundaram and Pruhs [11] analyzed a non-clairvoyant algorithm SETF (Shortest Elapsed Time First, which prefers jobs that have been processed the least) for a single processor, and they showed that it is $(1+\epsilon)$-speed $(1+\frac{1}{\epsilon})$-competitive for (unweighted) flow. Later, Bansal and Dhamdhere [4] generalized this result for weighted jobs, and showed that the algorithm WSETF is $(1+\epsilon)$-speed $(1+\frac{1}{\epsilon})$-competitive for weighted flow.

Multi-processor scheduling with arbitrary parallelizability. On a multi-core chip that provides $m \geq 1$ identical processors, some jobs might be processed faster when using several processors in parallel, while others might be inherently sequential. In the literature, the degree of parallelizability of a job was modeled as follows (e.g., [9, 10, 14]): A job consists of a number of phases, each with an arbitrary size and arbitrary speedup function that specifies the amount of speedup when running the job on a given number of processors. A non-clairvoyant scheduler has no information about the phases in advance. For minimizing (unweighted) flow in such a multi-processor model, Edmonds [9] gave a $(2+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive algorithm, and Edmonds and Pruhs [10] later showed that another algorithm LAPS (Latest Arrival Processor Sharing, which shares the processing power equally among a constant fraction of latest-arrival jobs) is $(1+\beta+\epsilon)$-speed $O(\frac{1+\beta+\epsilon}{\beta\epsilon})$-competitive, for any $\epsilon > 0$ and $0 < \beta \leq 1$ [10] . Very recently, Zhu et al. [14] extended the latter result to weighted jobs and gave an algorithm WLAPS (Weighted LAPS) that is $(1+\epsilon)$-speed $8(1+\frac{1}{\epsilon})^2$-competitive for weighted flow.

New results on flow plus penalty. In this paper, we extend the non-clairvoyant results on flow-time scheduling to the rejection penalty model. In particular, we propose a simple job rejection policy RWE (Reject When weighted flow Equals penalty), which rejects an unfinished job when the weighted flow incurred equals the job penalty. Unlike previous job rejection policies [2, 8], RWE does not require any information on job size, and it can be used in different settings. We develop a rather general technique for analyzing RWE. In the single-processor setting, we show that WLAPS coupled with RWE is $(1+\epsilon)$-speed $12(1+\frac{1}{\epsilon})^2$-competitive for minimizing total user cost; in the multi-processor setting, the competitive ratio becomes $20(1+\frac{1}{\epsilon})^2$. For the special case when jobs must all be completed (i.e., with infinite penalty), our new algorithm behaves exactly as WLAPS. The first two columns of Table 1 summarize the results on weighted flow and user cost (weighted flow plus penalty).

Dynamic speed scaling and energy. The above results assume that processors are running at a fixed speed. We show that RWE also works well in the dynamic speed scaling model, in which the scheduler can manage the energy consumption by scaling the processor speed dynamically (see [1] for a survey). Specifically, the processor speed can vary between 0 and a maximum speed T, and its rate of energy consumption (i.e., power) P increases with the speed s according to a certain given power function, say, $P(s) = s^3$. In this model, a scheduler has to determine dynamically which job and at what speed to execute.

Tradeoff between user cost and energy. The past few years have witnessed several online results on optimizing the tradeoff between flow and energy under the dynamic speed scaling model (see [1] for survey). Chan et al. [8] have also extended the study of rejection penalty to the dynamic speed scaling model and considered the tradeoff between user cost and energy consumption, but their result is limited to the clairvoyant setting. In particular, they gave a clairvoyant algorithm that is $(1+\epsilon)^2$-speed $O((1+\frac{1}{\epsilon})^2)$-competitive for minimizing total user cost plus energy on a single processor [8]. In this context, a λ-speed processor (where $\lambda > 1$) means that, given power $P(s)$, it can run at speed $\lambda \cdot s$. In this paper, we use RWE to obtain a non-clairvoyant algorithm RAW which is $(1+\epsilon)$-speed $36(1+\frac{1}{\epsilon})^2$-competitive for minimizing total user cost plus energy (see Table 1).

Multi-processor result. We further show that RWE also leads to a new non-clairvoyant algorithm for minimizing user cost plus energy on $m > 1$ processors, where each processor can scale its speed independently and jobs comprise phases with different degrees of parallelizability. The only past relevant work was done by Chan, Edmonds and Pruhs [6], who considered scheduling unweighted jobs non-clairvoyantly on processors with power function $P(s) = s^\alpha$ for any $\alpha > 1$, and maximum speed $T = \infty$. The objective is to minimize total flow plus energy, and jobs must be all completed. They showed a strong lower bound of $\Omega(m^{(\alpha-1)/\alpha^2})$ on the competitive ratio. In view of this lower bound, they assume that jobs satisfy two properties: (1) They do not have side effect, i.e., the execution of a job does not affect anything external to itself, so multiple copies of a job can run simultaneously. (2) They are checkpointable, i.e., each copy can save its state periodically and then restart each copy from the point of execution of the copy that made the most progress. Then Chan, Edmonds and Pruhs [6] were able to extend LAPS to a new algorithm MultiLAPS which is $O(2^\alpha \log m \frac{\alpha^2}{\log \alpha})$-competitive for minimizing (unweighted) flow plus energy, and showed that any algorithm for such checkpointable jobs is $\Omega(\log^{1/\alpha} m)$-competitive. Roughly speaking, previous speed scaling work scales the speed such that flow and energy are incurred at the same rate, yet MultiLAPS needs to run two times faster, leading to the multiplicative factor of 2^α in the competitive ratio.

We extend the above result to the rejection penalty model, and more interestingly, allowing weighted jobs and arbitrary power function $P(s)$. We give an algorithm MultiRAW which uses RWE as job rejection policy and is $2(1+\epsilon)$-speed $12(\log m + 2)(1+\frac{1}{\epsilon})^2$-competitive for minimizing total user cost plus energy. To analyze MultiRAW, we need to restrict the total processor speed of the optimal offline algorithm to follow some function depending on total weight of unfinished jobs, and show that such a restriction does not increase the competitive ratio by a constant factor. To this end, we generalize the offline transformation algorithm LLB (Latest Lag Behind) first given in [7] for single processor, and allow it to transform an offline algorithm that would run multiple different jobs simultaneously on multiple processors. The transformed offline algorithm may also run different jobs at the same time using time sharing and always has the processor speed following the required function, and we show that its weighted flow is at most the weighted flow plus energy of the original algorithm.

Organization of the paper. The following discussion focuses on the results in the dynamic speed scaling model. Note that the results in the fixed speed model would be shown as special cases. Section 2 defines the models, problems and notations formally. Results on single processor and multi-processors are given in Sections 3 and 4, respectively.

2. FORMAL PROBLEM DEFINITIONS AND NOTATIONS

We study job scheduling on a chip containing $m \geq 1$ identical processors. Jobs are arriving online, where each job j has a release time $r(j)$, a size $p(j)$, a weight $w(j)$ and a rejection penalty $c(j)$. Jobs are *non-clairvoyant*, meaning that the size of a job is unknown until it is completed. Preemption and migration are allowed and free.

Each processor can run independently at any speed $s \in [0, T]$ (where T is the maximum speed of the processor which may be ∞). The rate of energy usage is given by an arbitrary power function $P(s)$. Similar to [3], we assume that $P(0) = 0$, and P at all speeds in $[0, T]$ is defined, strictly increasing, nonnegative, continuous, strictly convex and differentiable. As shown in [3], it is possible to use such a power function P to emulate any arbitrary power function with an arbitrarily small increase in the competitive ratio. Let $Q(y) = \min\{P^{-1}(y), T\}$. Note that Q is monotonically increasing and concave. E.g., if $P(s) = s^\alpha$ for some $\alpha > 1$, then $Q(x) = x^{1/\alpha}$.

In the single processor setting, each job is processed by at most one processor and its processing rate is always the speed of the processor times the fraction of the processor assigned to this job. The processing rate is more complicated in the multi-processor setting due to the varying parallelizability of the job. In particular, we consider each job as a sequence of $q(j)$ phases $\langle j^1, j^2, \ldots, j^{q(j)}\rangle$. Each phase j^k is an ordered pair $\langle p(j^k), \Gamma_{j^k}\rangle$, where $p(j^k)$ is the amount of work in the phase and Γ_{j^k} is a *speed-up function* specifying the degree of parallelism of the phase. More precisely, $\Gamma_{j^k}(y)$ represents the rate at which work in the phase j^k is processed when using y processors running at speed 1. If the y processors are running at speed s, then the work is processed at rate $\Gamma_{j^k}(y) \cdot s$. Following the previous work (e.g., [6, 9, 10]), we assume that each speedup function Γ is non-negative, monotonically increasing and sublinear, i.e., $\frac{\Gamma(y)}{y} \geq \frac{\Gamma(y')}{y'}$ for any $y \leq y'$. We assume that for any phase, its speed-up function Γ satisfies that $\Gamma(y) = y$ for $y \in [0, 1]$. This assumption corresponds to the fact that when a phase is processed by time-sharing on a $y \leq 1$ fraction of a processor, its processing rate should be y times the speed of the processor.

In the non-clairvoyant setting, we assume that the size and speedup function of each phase is not known to the online algorithm.

In the speed scaling model, the objective is to minimize the sum of the total weighted flow time, total rejection penalty and total energy usage. We call it the *cost* of a schedule. Let OPT be the optimal offline algorithm that always minimizes the cost. We analyze our algorithms when they are given faster processors. Precisely, a y-speed processor runs at speed sy when the rate of energy usage is $P(s)$.

Previous definitions. Our results make use of previous work like WLAPS and AJW (e.g., [7]). We review the necessary definitions. We say that a job is *active* at time t if it has arrived but has not yet finished or rejected by time t. Throughout this paper, we denote $n_a(t)$ as the number of active job in the online algorithm at time t and $w_a(t)$ be their total weight.

DEFINITION 1 (β-ADJUSTED WEIGHT). *Let $\beta \in (0, 1]$ be a parameter. Consider any time t. Let $j_1, j_2, \ldots, j_{n_a(t)}$ be the active jobs in the online algorithm ordered in increasing order of arrival times. Let τ be the largest integer such that the latest arrived jobs $\{j_\tau, j_{\tau+1}, \ldots, j_{n_a(t)}\}$ have total weight at least $\beta w_a(t)$. Then, the* β-adjusted weight *of j_i at time t, denoted $w'_\beta(j_i, t)$ (or simply $w'(j_i, t)$) when β is clear in context), is defined as follows:*

$$w'_\beta(j_i, t) = \begin{cases} 0 & \textit{if } i < \tau \\ \beta w_a(t) - \sum_{i=\tau+1}^{n_a(t)} w(j_i) & \textit{if } i = \tau \\ w(j_i) & \textit{if } i > \tau \end{cases}$$

DEFINITION 2 (JOB ASSIGNMENT POLICY WLAPS). *WLAPS(β) is parameterized by a constant $\beta \in (0, 1]$. It assumes all processors are running at the same speed. At any time t, WLAPS shares the processors among all active jobs proportional to their adjusted weights at time t, i.e., each job j is assigned a fraction $\frac{w'(j,t)}{\beta w_a(t)}$ of the processors.*

DEFINITION 3 (SPEED SCALING POLICY AJW). *At any time t, AJW sets the speed of each processor to $s(t) = Q(\frac{w_a(t)}{m})$. Intuitively, the speed $s(t)$ ensures that the total rate of energy usage equals $w_a(t)$, or $s(t) = T$ if $mP(T) < w_a(t)$.*

3. SINGLE PROCESSOR RESULTS

In this section, we propose a simple job rejection policy RWE (Reject When weighted flow Equals penalty). Combining RWE with WLAPS, we can obtain an O(1)-competitive algorithm for minimizing weighted flow plus penalty on a fixed-speed processor. Below we show a more general result on combining RWE with WLAPS and AJW to obtain an O(1)-competitive algorithm for minimizing weighted flow plus penalty plus energy under the dynamic speed scaling model. RWE is defined as follows.

DEFINITION 4 (JOB REJECTION POLICY RWE). *RWE rejects a job j if it is not completed by time $r(j) + \frac{c(j)}{w(j)}$.*

Algorithm RAW. Let $\beta \in (0, 1]$ be a constant. RAW assumes a $(1+\epsilon)$-speed processor where $\epsilon > 0$ can be any real. RAW assigns jobs to the processor by WLAPS, i.e., sharing the processor among all active jobs proportional to their adjusted weights. RAW scales the processor speed to $(1 + \epsilon) \cdot Q(w_a(t))$ (so the rate of energy usage is $w_a(t)$). RAW rejects jobs according to RWE.

We call the algorithm RAW to stand for RWE-AJW-WLAPS. Our main result is the following.

THEOREM 1. *Let $\beta = \frac{\epsilon}{2(1+\epsilon)}$. RAW is $(1+\epsilon)$-speed $36(1+\frac{1}{\epsilon})^2$-competitive for minimizing weighted flow plus penalty plus energy.*

To prove Theorem 1, let OFF be the offline algorithm that minimizes the cost under the condition that OFF scales the processor speed by AJW, and OFF rejects a job j at time $r(j)$ if OPT rejects j. It is known that the cost of OFF is at most twice of OPT when they do not reject jobs [7]. Below we show that this relationship remains valid even if they reject jobs.

LEMMA 2. *The cost of* OFF *is at most twice the cost of* OPT.

PROOF. For each job j, recall that OFF rejects j when j arrives if OPT rejects j. Then, OFF schedules the remaining jobs by the algorithm LLB [7], which uses AJW to scale the processor speed. [7] shows that the total weighted flow time plus energy usage of LLB is at most twice that of OPT. Since the total penalty of OFF is the same as that of OPT, the cost of OFF is at most twice that of OPT. $\Box$

Hence, it suffices to analyze RAW against OFF, which would incur an extra factor of two in the competitive ratio. At any time t, let $q_a(j,t)$ be the remaining work of j in RAW and let $q_o(j,t)$ be that in OFF. Note that $q_a(j,t)$ (resp., $q_o(j,t)$) becomes 0 once RAW (resp., OFF) rejects j. We are interested in the progress of the jobs that are not rejected by OFF.

DEFINITION 5 (LAGGING WORK). *At any time t, for any job j, the amount of* lagging work *of j is defined as*

$$x(j,t) = \begin{cases} 0 & \text{if OFF rejects } j \text{ at } r(j) \\ \max\{q_a(j,t) - q_o(j,t), 0\} & \text{otherwise} \end{cases}$$

Intuitively, $x(j,t)$ is the amount of "useful" work that RAW is lagging behind OFF. In particular, if j is already rejected by OFF, then all work remaining in RAW is not useful.

At any time t, let $R(t)$ be the set of jobs being processed by RAW. Recall that the total adjusted weight of jobs in $R(t)$ is $\beta w_a(t)$, where $w_a(t)$ is the total weight of all active jobs. Let $L(t) \subseteq R(t)$ be those jobs in $R(t)$ such that $x(j,t) > 0$. We call $L(t)$ the lagging jobs. We denote $\phi(t) = \sum_{j \in L(t)} w'(j,t)$. Then Theorem 1 follows from the following two lemmas.

LEMMA 3 (LOWER BOUND OF OFF). $\int_0^\infty (\beta w_a(t) - \phi(t))dt \le W_o + C_o$, *where W_o and C_o are the total weighted flow and the total rejection penalty of* OFF, *respectively.*

LEMMA 4 (UPPER BOUND OF RAW). $\int_0^\infty w_a(t)\ dt \le \frac{1}{\beta^2}\int_0^\infty ((\beta w_a(t) - \phi(t)) + \beta w_o(t))dt$, *where $w_o(t)$ is the total weight of active jobs in* OFF *at time t.*

Before proving Lemmas 3 and 4, we show how to use these two lemmas to prove Theorem 1.

PROOF. (*Proof of Theorem 1*) Recall that that $\beta < 1/2$. By Lemmas 3 and 4, the total weighted flow time of RAW is at most $\frac{1}{\beta^2}(W_o + C_o + \int_0^\infty \beta w_o(t)dt) = (2(1+\frac{1}{\epsilon}))^2((1+\beta)W_o + C_o) \le 6(1+\frac{1}{\epsilon})^2 \cdot (W_o + C_o)$. Note that for each job j rejected by RWE, the weighted flow time incurred is $\frac{c(j)}{w(j)} \cdot w(j) = c(j)$. Hence, the total penalty of RAW is at most its total weighted flow time. Furthermore, by running AJW, RAW has energy usage at most its weighted flow time. In summary, the cost of RAW is at most $18(1+\frac{1}{\epsilon})^2$ times the cost of OFF. The latter is at most twice the cost of OPT (by Lemma 2). Theorem 1 follows. $\Box$

PROOF. (*Proof of Lemma 3*) Consider any time t. Note that $\beta w_a(t) - \phi(t)$ is the total weight of jobs in $R(t) \setminus L(t)$, which are the jobs j being processed by RAW with $x(j,t) = 0$. Note that $x(j,t) = 0$ only if j is rejected by OFF at $r(j)$ or $q_a(j,t) \le q_o(j,t)$ at time t. Let C be the set of jobs rejected by OFF. Let $C(t) = C \cap R(t) \setminus L(t)$ be those jobs in C that are still active in RAW at time t and let $N(t) \subseteq R(t) \setminus L(t)$ be those jobs j not rejected by OFF but $q_a(j,t) \le q_o(j,t)$ at time t. Then,

$$\int_0^\infty (\beta w_a(t) - \phi(t))dt = \int_0^\infty \sum_{j \in C(t)} w(j)dt + \int_0^\infty \sum_{j \in N(t)} w(j)dt.$$

Note that for each job j, j remains in RAW for at most $\frac{c(j)}{w(j)}$ units of time and incurs a weighted flow time of at most $c(j)$. Hence, $\int_0^\infty \sum_{j \in C(t)} w(j)dt \le \sum_{j \in C} c(j) = C_o$.

For each job j in $N(t)$, j is not completed by OFF at time t. Hence, $\int_0^\infty \sum_{j \in N(t)} w(j)dt \le \int_0^\infty w_o(t)dt \le W_o$, where $w_o(t)$ is the total weight of jobs in OFF at time t. $\Box$

PROOF. (*Proof of Lemma 4*) We use a potential function analysis. Let $j_1, j_2, \ldots, j_{n_a(t)}$ be the active jobs in RAW at time t arranged in increasing order of arrival times. We denote $h(j_i) = \sum_{k=1}^{i} w(j_i)$ as the total weight of active jobs arrived no later than j_i. Recall that $x(j,t)$ is the lagging work of j at time t. Then, we define

$$\Phi(t) = \gamma \sum_{i=1}^{n_a(t)} f(h(j_i)) \cdot x(j_i, t),$$

where $\gamma = \frac{1}{\beta(1+\epsilon)}$ and $f(h) = h/Q(h)$. Note that $f(h)$ is non-decreasing.[1] We will show that Φ satisfies the following three conditions.

- *Boundary condition*: $\Phi = 0$ before the first job arrives and after all jobs are finished.
- *Discrete event condition*: Φ does not increase at job arrival, completion or rejection.
- *Running condition*: At any other time t,

$$w_a(t) + \frac{d\Phi(t)}{dt} \le \frac{1}{\beta^2}((\beta w_a(t) - \phi(t)) + \beta w_o(t)) .$$

Then, by integrating these conditions over time, the lemma follows. The boundary condition is true as $n_a(t) = 0$ before any job is released and after all jobs are finished. When a job j arrives, no matter whether OFF rejects j, $x(j,t) = 0$, so Φ does not increase. When j is completed by OFF, $x(j,t)$ and Φ do not change. When j is completed or rejected by RAW, the term corresponding to j disappears and other terms may only decrease, so Φ does not increase. Thus, the discrete event condition is true. For the running condition, Lemma 5 below shows that $\frac{d\Phi}{dt} \le \frac{1-2\beta}{\beta}\max\{w_a(t), w_o(t)\} - \frac{1}{\beta^2}(1-\beta)\phi(t)$. Note that $\frac{1-2\beta}{\beta}\max\{w_a(t), w_o(t)\}$ is no greater than $\frac{1}{\beta}\max\{w_a(t), w_o(t)\} - 2w_a(t)$, and $\frac{1}{\beta}\phi(t) \le \frac{1}{\beta}\beta w_a(t) \le w_a(t)$. Therefore,

$$\begin{aligned}
& w_a(t) + \frac{d\Phi(t)}{dt} \\
\le\ & w_a(t) + \frac{1-2\beta}{\beta}\max\{w_a(t), w_o(t)\} - \frac{1}{\beta^2}(1-\beta)\phi(t) \\
\le\ & w_a(t) + \frac{1}{\beta}\max\{w_a(t), w_o(t)\} - 2w_a(t) - \frac{1}{\beta^2}\phi(t) + w_a(t) \\
\le\ & \frac{1}{\beta^2}\left[\beta \max\{w_a(t), w_o(t)\} - \phi(t)\right] \\
\le\ & \frac{1}{\beta^2}((\beta w_a(t) - \phi(t)) + \beta w_o(t)).
\end{aligned}$$

$\Box$

[1]Since Q is concave, for any $\lambda \in [0,1]$ and $h \ge 0$, $(1-\lambda)Q(0) + \lambda Q(h) \le Q((1-\lambda)0 + \lambda h)$, i.e., $\lambda Q(h) \le Q(\lambda h)$. Hence, $f(\lambda h) = \frac{\lambda h}{Q(\lambda h)} \le \frac{h}{Q(h)} = f(h)$.

LEMMA 5. *At any time t without discrete events,* $\frac{d\Phi}{dt} \leq \frac{1-2\beta}{\beta} \cdot \max\{w_a(t), w_o(t)\} - \frac{1}{\beta^2}(1-\beta)\phi(t)$.

PROOF. We consider $\frac{d\Phi}{dt}$ as a combined effect due to the action of RAW and OFF. Let $\frac{d\Phi_a}{dt}$ and $\frac{d\Phi_o}{dt}$ be the rate of change of Φ due to RAW and OFF, respectively. Then $\frac{d\Phi}{dt} = \frac{d\Phi_a}{dt} + \frac{d\Phi_o}{dt}$.

For each active job j in RAW, if $j \in L(t)$, $x(j,t) > 0$ and $h(j) \geq (1-\beta)w_a(t)$. RAW processes j at a speed $(1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} \cdot Q(w_a(t))$. Hence, for each term $\gamma \cdot f(h(j)) \cdot x(j,t)$ where $j \in L(t)$, the term is decreasing at a rate at least

$$\begin{aligned} & \gamma \cdot \frac{h(j)}{Q(h(j))} \cdot (1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} Q(w_a(t)) \\ \geq\ & \gamma \cdot \frac{(1-\beta)w_a(t)}{Q((1-\beta)w_a(t))} \cdot (1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} Q(w_a(t)) \\ \geq\ & \gamma \frac{1-\beta}{\beta}(1+\epsilon)w'(j,t) = \frac{1-\beta}{\beta^2} w'(j,t) \end{aligned}$$

where the first inequality comes from that $f(h) = h/Q(h)$ is non-decreasing with h, and the second inequality comes from that Q is increasing. For each job $j \notin L(t)$, the term $\gamma \cdot f(h(j)) \cdot x(j,t)$ is non-increasing. Hence, $\frac{d\Phi_a}{dt} \leq -\frac{1-\beta}{\beta^2}\sum_{j\in L(t)} w'(j,t) = -\frac{1-\beta}{\beta^2}\phi(t)$.

For OFF, the worst case (in which $\frac{d\Phi_o}{dt}$ is the largest) occurs when it processes the job with maximum $h(j)$, which equals $w_a(t)$, using all its speed $s_o = Q(w_o(t))$. Then $\frac{d\Phi_o}{dt} \leq \gamma \cdot f(w_a(t)) \cdot s_o = \gamma \frac{w_a(t)}{Q(w_a(t))} \cdot Q(w_o(t))$. If $w_a(t) \geq w_o(t)$, $Q(w_a(t)) \geq Q(w_o(t))$ and $\frac{d\Phi_o}{dt} \leq \gamma w_a(t)$. Otherwise, $w_a(t) < w_o(t)$. Since f is non-decreasing, $f(w_a(t)) \leq f(w_o(t))$. Hence, $\frac{d\Phi_o}{dt} \leq \gamma f(w_o(t)) \cdot s_o = \gamma w_o(t)$. Combining the two cases, we have $\frac{d\Phi_o}{dt} \leq \gamma \cdot \max\{w_a(t), w_o(t)\}$. Notice that $\beta = \frac{\epsilon}{2(1+\epsilon)}$ and $\frac{1}{1+\epsilon} = 1 - 2\beta$. Hence, $\gamma \max\{w_a(t), w_o(t)\} = \frac{1}{\beta(1+\epsilon)} \cdot \max\{w_a(t), w_o(t)\} = \frac{1-2\beta}{\beta}\max\{w_a(t), w_o(t)\}$ and the lemma follows. □

Remarks on fixed speed setting. Since the fixed speed setting is a special case of the arbitrary power function setting (where energy usage of any algorithm is zero), the previous result immediately implies that RAW is $36(1+\frac{1}{\epsilon})^2$-competitive using a $(1+\epsilon)$-speed processor. Yet we can tighten the analysis to achieve a better competitive ratio. In particular, we can directly compare RAW and OPT and show lemmas similar to Lemma 3 and 4, which can upper bound the total weighted flow of RAW to be at most $6(1+\frac{1}{\epsilon})^2$ times the cost of OPT. The total rejection penalty of RAW is at most its weighted flow. The following theorem follows.

THEOREM 6. *For the fixed speed setting, RAW is $(1+\epsilon)$-speed $12(1+\frac{1}{\epsilon})^2$-competitive for minimizing weighted flow plus penalty.*

4. MULTI-PROCESSOR RESULTS

We consider scheduling on $m > 1$ processors in the speed scaling model. We give an algorithm MultiRAW which is $2(1+\epsilon)$-speed $12(1+\frac{1}{\epsilon})^2(\log m+2)$-competitive for minimizing weighted flow plus penalty plus energy. Note that MultiRAW follows the job rejection policy RWE given in Section 3. The fixed speed model is treated as a special case; we can modify the result of MultiRAW to get an $(1+\epsilon)$-speed $20(1+\frac{1}{\epsilon})^2$-competitive algorithm for weighted flow plus penalty. For simplicity, we first assume that the number of processors m is a power of two. We will explain how to remove this assumption later.

Our algorithm is built on MultiLAPS proposed by [6], in which they consider unweighted jobs that cannot be rejected and the power function is $P(s) = s^\alpha$ without maximum speed bound. Their objective is to minimize the total flow time plus energy usage.

Algorithm MultiLAPS [6]. Let $\beta \in (0,1]$ be a constant and $n_a(t)$ be the number of active jobs in MultiLAPS at time t. MultiLAPS divides the m processors into $\log m$ groups such that for $\ell = 1, \ldots, \log m$, the ℓ-th group consists of $m_\ell = 2^{-\ell}m$ processors each running at speed $2(\frac{n_a(t)}{m_\ell})^{1/\alpha}$. Each group runs independently and processes the $\lceil \beta n_a(t) \rceil$ jobs with the latest arrival times by sharing the processors within the group equally among the jobs. Note that each of these jobs is duplicated into $\log m$ copies and processed simultaneously by the $\log m$ groups. Since the jobs are checkpointable, at any time, the processing rate of a job is the maximum processing rate among all the copies of the job in the $\log m$ groups.

Algorithm MultiRAW. To handle weighted jobs, rejection penalty, and arbitrary power functions, we extend the algorithm MultiLAPS as follows.

- For the ℓ-th group, we set the speed of each processor to $s_\ell = 2(1+\epsilon)Q(\frac{w_a(t)}{m_\ell})$; recall that $Q(y) = \min\{P^{-1}(y), T\}$, and $w_a(t)$ is the total weight of the active jobs at time t.
- We share the processors within a group proportional to the adjusted weight of the jobs. Hence, a job j is processed by $\frac{w'(j,t)}{\beta w_a(t)} \cdot m_\ell$ processors in the ℓ-th group.
- (Job rejection policy RWE) We reject a job j if it is not finished by time $r(j) + \frac{c(j)}{w(j)}$.

Our main result is the following theorem. Sections 4.1 to 4.5 are devoted to proving this theorem.

THEOREM 7. *Let $\epsilon > 0$ be any real and $\beta = \frac{\epsilon}{2(1+\epsilon)}$. MultiRAW is $2(1+\epsilon)$-speed $12(1+\frac{1}{\epsilon})^2(\log m+2)$-competitive for minimizing weighted flow plus penalty plus energy.*

4.1 Restricting input instances and offline algorithm

Canonical instances. We first show that it suffices to consider some specific input instances. Recall that jobs may have varying parallelizability and the parallelizability of a phase is given by a speedup function Γ. We call Γ *parallel up to σ processors* if $\Gamma(y) = y$ for all $y \leq \sigma$ and $\Gamma(y) = \sigma$ for $y > \sigma$. We call an instance *canonical* if each job phase is parallel up to σ processors for some $\sigma \in [1, m]$, where σ may be different for different phases.

LEMMA 8. *For any input instance I, we can transform I into a canonical instance such that the cost of MultiRAW does not change and the cost of* OPT *may only decrease.*

Canonical instances were first introduced in [6,14], and Lemma 8 can be proven in a similar way as in [6, 14]. In the following, we consider canonical instances only.

Restricted offline algorithm. To prove Theorem 7, we need to compare MultiRAW against the optimal offline algorithm OPT. Without loss of generality, OPT rejects jobs at their release time. It is sometimes easier to compare MultiRAW against an offline algorithm that rejects the same set of jobs as OPT but works on a single processor with maximum speed $m \cdot T$. Let OFF be such an offline algorithm that minimizes the weighted flow alone under the condition that the single processor always runs at speed $m \cdot Q(\frac{w_o(t)}{m})$, where $w_o(t)$ is the total weight of active jobs in OFF. Note that OFF does not have energy concern in its objective.

LEMMA 9. *The weighted flow of* OFF *is at most the weighted flow plus energy of* OPT.

To prove Lemma 9, we generalize the algorithm LLB (Latest Lag Behind) given in [7] to transform OPT (which minimizes weighted flow plus energy of the jobs not rejected on m processors) to an offline algorithm LLB' on a single processor with maximum speed $m \cdot T$ that at any time t, follows the speed $m \cdot Q(\frac{w_b(t)}{m})$, where $w_b(t)$ is the total weight of active jobs in LLB'. A new feature of LLB' is the ability to handle the case that OPT can run multiple jobs by time sharing at the same time.

Algorithm $\mathbf{LLB'}$. Consider any time t. Let $n_b(t)$ be the number of active jobs in LLB'. For any job j, let $p_b(j,t)$ and $p_o(j,t)$ be the work done on j up to time t by LLB' and OPT, respectively. Furthermore, let $d(j,t) = p_o(j,t) - p_b(j,t)$. We say a job j is *lagging* at time t if $d(j,t) > 0$. Let $y(j)$ be the latest time when j has become lagging; if j is non-lagging, let $y(j) = t$. We denote the active jobs in LLB' at time t as $j_1, j_2, \ldots, j_{n_b(t)}$, arranged in increasing order of $y(j_i)$, i.e., $y(j_1) \le y(j_2) \le \cdots \le y(j_{n_b(t)})$. Let ℓ be the number of lagging jobs ($0 \le \ell \le n_b(t)$). Let s_o be the total speed of the m processors in OPT at time t. Let $s_x \le s_o$ be the total speed OPT assigns to all jobs j with $d(j,t) = 0$. LLB' sets its speed $s_b = m \cdot Q(\frac{w_b(t)}{m})$, and focuses on the job j_a defined to be j_ℓ if $\ell > 0$, and $j_{n_b(t)}$ otherwise. Details are as follows:

- If OPT is not running any job j_i with $d(j_i,t) = 0$, then LLB' runs j_a with speed s_b.
- If OPT is processing some jobs j_i with $d(j_i,t) = 0$, but $s_x \le s_b$ (i.e., LLB' has enough speed to prevent all of them from becoming lagging), then LLB' follows OPT's speed on each of those jobs and runs j_a with the remaining speed $s_b - s_x$.
- Otherwise ($s_b < s_x$), LLB' follows OPT's speed for each of those jobs j_i, in descending order of the index i, until there is some job j which LLB' cannot follow OPT's total speed on j. LLB' then assigns all of its remaining speed to j.

Roughly speaking, LLB' attempts to catch up with the progress of OPT; it gives priority to the job that has become lagging most recently, while trying to avoid creating more new lagging jobs. We can show that the weighted flow of LLB' is at most the weighted flow plus energy of OPT. Its proof is given in Appendix B. Since OFF minimizes weighted flow, Lemma 9 follows.

4.2 Lower bound on processing of MultiRAW

At any time t, consider any active job j in MultiRAW. We define $\sigma(j,t)$ such that the next available work of j belongs to a phase that is parallel up to $\sigma(j,t)$ processors. We call $\sigma(j,t)$ the *saturated number* of j at time t. Intuitively, any processor allocated to j beyond $\sigma(j,t)$ is wasted and cannot further increase the processing rate.

At any time t, let $R(t)$ be the set of jobs j with positive adjusted weight, i.e., $w'(j,t) > 0$. Note that $R(t)$ is the set of jobs being processed by MultiRAW and the processing rate of a job $j \in R(t)$ is the maximum of its processing rate in the $\log m$ groups. We classify the work of j into two types as follows. We label the work as *unsaturated* if it is processed by no more processors than its saturation number in *all* groups; and label the work as *saturated* otherwise. We call a job $j \in R(t)$ saturated at time t if MultiRAW is processing its saturated work, and call j unsaturated otherwise. Intuitively, by running j on $\log m$ groups, MultiRAW guarantees at least one group has a sufficient processing rate on j.

LEMMA 10. *At any time t, consider any job $j \in R(t)$. If j is unsaturated, the processing rate on j is at least $(1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} \cdot m \cdot Q(\frac{w_a(t)}{m})$. If j is saturated, the processing rate on j is at least $(1+\epsilon) \cdot \sigma(j,t) \cdot Q(\frac{w'(j,t)}{\sigma(j,t)})$.*

PROOF. If j is unsaturated, consider the processing rate of j by the first group, which has $m/2$ processors each running at speed $2(1+\epsilon)Q(\frac{w_a(t)}{m/2})$. The job j receives $\frac{w'(j,t)}{\beta w_a(t)}$ fraction of the processors. Since j is unsaturated, the processing rate on j equals $m/2 \cdot \frac{w'(j,t)}{\beta w_a(t)} \cdot 2(1+\epsilon)Q(\frac{w_a(t)}{m/2})$. Since Q is non-decreasing and $\frac{w_a(t)}{m/2} \ge \frac{w_a(t)}{m}$, we obtain the desired bound.

If j is saturated, we let the k-th group be the group such that $m_k \frac{w'(j,t)}{\beta w_a(t)} \le \sigma(j,t) < 2m_k \frac{w'(j,t)}{\beta w_a(t)}$, where m_k is the number of processors in the group. Note that k must exist since $\sigma(j,t) \ge 1 = m_{\log m} \ge m_{\log m} \frac{w'(j,t)}{\beta w_a(t)}$. The processing rate on j by this group equals $m_k \frac{w'(j,t)}{\beta w_a(t)} \cdot 2(1+\epsilon)Q(\frac{w_a(t)}{m_k}) \ge (1+\epsilon)\sigma(j,t)Q(\frac{w_a(t)}{m_k})$. Since $\frac{w_a(t)}{m_k} \ge \frac{w'(j,t)}{\beta\sigma(j,t)} \ge \frac{w'(j,t)}{\sigma(j,t)}$, we obtain the desired bound. □

4.3 Bounding weighted flow of MultiRAW

To bound the weighted flow of MultiRAW, we identify a subset of jobs such that the weighted flow of MultiRAW is at most a constant times the total weighted flow of these jobs in MultiRAW. Then, we upper bound this cost by the weighted flow of OFF and the cost of OPT in Section 4.4, and the desired result follows.

Now, we compare MultiRAW against OFF. For any job j and time t, let $q_a(j,t)$ and $q_o(j,t)$ be the remaining amount of *unsaturated* work of j in MultiRAW and OFF, respectively. In particular, $q_a(j,t)$ (resp., $q_o(j,t)$) is zero if j has been rejected by MultiRAW (resp., OFF) by time t. Recall that OFF only rejects j at $r(j)$. We define the amount of lagging work of a job j, denoted $x(j,t)$, in the same way as in Definition 5 in Section 3.

Consider any time t. It is useful to have a detailed breakdown of the set $R(t)$ of jobs being processed by MultiRAW. We divide $R(t)$ into two sets $S(t)$ and $U(t)$ containing the saturated and unsaturated jobs, respectively. We further divide $U(t)$ into $L(t)$ and $N(t)$ which contain jobs with $x(j,t) > 0$ and $x(j,t) = 0$, respectively. We call $L(t)$ the lagging jobs and $N(t)$ the non-lagging jobs. Recall that the total adjusted weight of jobs in $R(t)$ is $\beta w_a(t)$. Let $\phi(t) = \sum_{j \in L(t)} w'(j,t)$. Similar to Lemma 4 in the single processor setting, we try to bound the total weighted flow time of MultiRAW by weighted flow time incurred due to jobs in $R(t) \setminus L(t)$.

LEMMA 11 (WEIGHTED FLOW OF MULTIRAW).

$$\int_0^\infty w_a(t)dt \le \frac{1}{\beta^2}\int_0^\infty ((\beta w_a(t) - \phi(t)) + \beta w_o(t))\, dt,$$

where $w_o(t)$ is the total weight of active jobs in OFF at time t.

The proof is similar to that of Lemma 4. In particular, by Lemma 10, the processing rate of each unsaturated job j in MultiRAW is at least $(1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} \cdot m \cdot Q(\frac{w_a(t)}{m})$. On the other hand, the processing rate of an unsaturated job j in OFF is at most $m \cdot Q(\frac{w_o(t)}{m})$. Hence, by redefining the potential function $\Phi(t)$ to $\gamma \sum_{i=1}^{n_a(t)} f(\frac{h(j_i)}{m}) \cdot x(j_i,t)$, we can prove a lemma identical to Lemma 5 (see Appendix A). Then we can prove Lemma 11 in the same way as Lemma 4.

4.4 Bounding non-lagging jobs and saturated jobs

In the following, we show that $\int_0^\infty (\beta w_a(t) - \phi(t))dt$ can be bounded by the weighted flow of OFF and the cost of OPT. Let C be set of jobs rejected by OPT, which is also the set of jobs rejected by OFF. Let $C(t) = C \cap (S(t) \cup N(t))$. Then, $\beta w_a(t) - \phi(t) = \sum_{j \in C(t)} w'(j,t) + \sum_{j \in N(t) \setminus C(t)} w'(j,t) + \sum_{j \in S(t) \setminus C(t)} w'(j,t)$.

We further break down the proof into two parts. Let W^*, E^* and C^* be the weighted flow, energy and penalty of OPT, respectively. Let W_o be the weighted flow of OFF.

LEMMA 12.
$$\int_0^\infty \sum_{j\in C(t)} w'(j,t) + \sum_{j\in N(t)\setminus C(t)} w'(j,t)dt \le W_o + C^*$$

PROOF. For each $j \in C(t)$, j remains in MultiRAW for at most $c(j)/w(j)$ units of time, incurring a weighted flow of at most $c(j)$. Hence, $\int_0^\infty \sum_{j\in C(t)} w(j)dt \le \sum_{j\in C} c(j) = C^*$. For each $j \in N(t) \setminus C(t)$, j is unfinished by OFF at time t. Hence, $\int_0^\infty \sum_{j\in N(t)\setminus C(t)} w(j)dt \le \int_0^\infty w_o(t)dt = W_o$, where $w_o(t)$ is the total weight of active jobs in OFF at time t. The lemma follows by observing that $w'(j,t) \le w(j)$ for any job j at any time t. □

LEMMA 13. $\displaystyle\int_0^\infty \sum_{j\in S(t)\setminus C(t)} w'(j,t)dt \le W^* + E^*.$

PROOF. Consider any job j not rejected by OPT. Let $\Delta(j)$ be the union of all saturated work in j. We divide the saturated work $\Delta(j)$ into infinitesimal pieces $\{x_1, x_2, \dots\}$ such that (1) within a piece of work, the saturation number remains the same; and (2) MultiRAW processes each piece continuously at a fixed rate, and so as OPT; and (3) there is no job arrival, rejection or completion during the processing of a piece. Each piece is infinitesimal. For any piece $x \in \Delta(j)$, we let $\sigma(x)$ be its saturation number. Let $s(x)$ and $s^*(x)$ be the processing rate on x by MultiRAW and OPT, respectively. Let $w'(x)$ be the adjusted weight of j when x is being processed by MultiRAW. Let $p(x)$ be the size of x. Let S^* be the set of jobs that are not rejected by OPT. Then,
$$\int_0^\infty \sum_{j\in S(t)\setminus C(t)} w'(j,t)dt = \sum_{j\in S^*}\sum_{x\in\Delta(j)} \frac{p(x)}{s(x)} w'(x)$$

Since x is a piece of saturated work, by Lemma 10, $s(x) \ge (1+\epsilon)\sigma(x)Q(\frac{w'(x)}{\sigma(x)}) > \sigma(x)Q(\frac{w'(x)}{\sigma(x)})$. Hence, $\frac{p(x)}{s(x)}w'(x) \le p(x)\frac{w'(x)/\sigma(x)}{Q(w'(x)/\sigma(x))} \le p(x)\frac{w(j)/\sigma(x)}{Q(w(j)/\sigma(x))}$. The last inequality comes from the fact that $\frac{y}{Q(y)}$ is increasing with y (as Q is nonnegative and concave) and $w'(x) \le w(j)$.

Consider the same piece of work x. The weighted flow incurred by x in OPT is $\frac{p(x)}{s^*(x)}w(j)$. OPT processes x at the rate $s^*(x)$; the most energy efficient way is to use $\sigma(x)$ processors each running at speed $\frac{s^*(x)}{\sigma(x)}$. The rate of energy usage is $\sigma(x)\cdot P(s^*(x)/\sigma(x))$. Considering the cost in OPT, we have
$$W^* + E^* \ge \sum_{j\in S^*}\sum_{x\in\Delta(j)} \left(\frac{p(x)}{s^*(x)}w(j) + \frac{p(x)}{s^*(x)}\cdot\sigma(x)\cdot P(\frac{s^*(x)}{\sigma(x)})\right)$$

It remains to show $\frac{w(j)}{s^*(x)} + \frac{\sigma(x)}{s^*(x)}\cdot P(\frac{s^*(x)}{\sigma(x)}) \ge \frac{w(j)/\sigma(x)}{Q(w(j)/\sigma(x))}$. To this end, we consider two cases. If $s^*(x) \le \sigma(x)Q(w(j)/\sigma(x))$, then $\frac{w(j)}{s^*(x)} \ge \frac{w(j)/\sigma(x)}{Q(w(j)/\sigma(x))}$; otherwise, we note that $\frac{P(y)}{y}$ is increasing with y.
$$\begin{aligned}\frac{\sigma(x)}{s^*(x)}\cdot P(\frac{s^*(x)}{\sigma(x)}) = \frac{P(s^*(x)/\sigma(x))}{s^*(x)/\sigma(x)} &\ge \frac{P(Q(w(j)/\sigma(x)))}{Q(w(j)/\sigma(x))}\\ &= \frac{w(j)/\sigma(x)}{Q(w(j)/\sigma(x))}\end{aligned}$$

The last equality follows from the fact that $\sigma(x)T \ge s^*(x) > \sigma(x)Q(w(j)/\sigma(x))$. Therefore, $T > Q(w(j)/\sigma(x))$ and we have $P(Q(w(j)/\sigma(x))) = w(j)/\sigma(x)$. □

4.5 Conclusion — Proof of Theorem 7

We are ready to prove Theorem 7.

PROOF. (*Proof of Theorem 7*) By Lemmas 12 and 13, we have $\int_0^\infty(\beta w_a(t) - \phi(t))dt \le W_o + W^* + E^* + C^*$. Together with Lemma 11 and that $\beta < \frac{1}{2}$, the weighted flow of MultiRAW is at most $\frac{1}{\beta^2}((1+\beta)W_o + W^* + E^* + C^*) \le \frac{1}{\beta^2}(1.5W_o + W^* + E^* + C^*)$, which by Lemma 9, is at most $\frac{2.5}{\beta^2}$ times the cost of OPT. The total energy usage of MultiRAW is at most $\log m$ times its weighted flow time. The rejection penalty of MultiRAW is at most its weighted flow time. Hence, MultiRAW is at most $(\frac{2.5}{\beta^2}(\log m + 2))$-competitive against OPT. Finally, putting $\beta = \frac{\epsilon}{2(1+\epsilon)}$, we conclude that MultiRAW is $10(1+\frac{1}{\epsilon})^2(\log m + 2)$-competitive.

So far, we have assumed that m is a power of two. If m is not a power of two, let $m_1 = \lceil m/2 \rceil$ and $m_i = \left\lceil (m - \sum_{j=1}^{i-1} m_j)/2 \right\rceil$, where m_i is the number of processors in the i-th group. We can show that $m_i \le 3m_{i+1}$. Repeating the argument of Lemma 10, we can show that each saturated job j has a processing rate at least $\frac{2}{3}(1+\epsilon)\cdot\sigma(j,t)\cdot Q(\frac{w'(j,t)}{\sigma(j,t)})$. The only consequence is that the right-hand-side of Lemma 13 is weakened to $1.5(W^* + E^*)$. The competitive ratio of MultiRAW is increased by a factor of $\frac{3}{2.5}$ and becomes $12(1+\frac{1}{\epsilon})^2(\log m + 2)$. □

Remarks on fixed speed setting. If all the processors have speed one and energy is not a concern, we can largely simplify the algorithm by running a single group with all the m processors. We assume that the online algorithm is given $(1+\epsilon)$-speed processors and we share the processors to the active jobs proportional to the adjusted weight $w'(j,t)$. Then, similar to Lemma 10, we can show that each unsaturated job is processed at a rate at least $\frac{w'(j,t)}{\beta w_a(t)}m(1+\epsilon)$ and each saturated job is processed at a rate at least $(1+\epsilon)\sigma(j,t)$. We can check that Lemmas 11, 12 and 13 remain true, where $E^* = 0$. Hence, the total weighted flow is at most $\frac{2.5}{\beta^2}$ times the cost of OPT. The rejection penalty of the online algorithm is at most its total weighted flow time. Note that $\beta = \frac{\epsilon}{2(1+\epsilon)}$. Hence, we have the following theorem.

THEOREM 14. *For the fixed speed setting, there is a $(1+\epsilon)$-speed $20(1+\frac{1}{\epsilon})^2$-competitive algorithm.*

Note that for each job j, we only process one copy of j. Hence, the results hold even if the jobs are non-checkpointable.

5. REFERENCES

[1] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

[2] N. Bansal, A. Blum, S. Chawla, and K. Dhamdhere. Scheduling for flow-time with admission control. In *Proc. ESA*, pages 43–54, 2003.

[3] N. Bansal, H. L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. SODA*, pages 693–701, 2009.

[4] N. Bansal and K. Dhamdhere. Minimizing weighted flow time. *ACM Transactions on Algorithms*, 3(4):39, 2007.

[5] H. L. Chan, J. Edmonds, T. W. Lam, L. K. Lee, A. Marchetti-Spaccamela, and K. Pruhs. Nonclairvoyant speed scaling for flow and energy. *Algorithmica*, 61(3):507–517, 2011.

[6] H. L. Chan, J. Edmonds, and K. Pruhs. Speed scaling of processes with arbitrary speedup curves on a multiprocessor. In *Proc. SPAA*, pages 1–10, 2009.

[7] 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.

[8] S. H. Chan, T. W. Lam, and L. K. Lee. Scheduling for weighted flow time and energy with rejection penalty. In *Proc. STACS*, pages 392–403, 2011.

[9] J. Edmonds. Scheduling in the dark. *Theor. Comput. Sci.*, 235(1):109–141, 2000.

[10] J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. In *Proc. SODA*, pages 685–692, 2009.

[11] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[12] R. Motwani, S. Phillips, and E. Torng. Nonclairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

[13] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. FOCS*, pages 374–382, 1995.

[14] J. Zhu, H. L. Chan and T. W. Lam. Non-clairvoyant weighted flow time scheduling on different multi-processor models. To appear in *Proc. WAOA*, 2011.

Appendix A: Lemma 5 for Multi-processor Setting

In this appendix, we consider $m > 1$ processors and prove Lemma 5 for the multi-processor setting. Recall that $j_1, j_2, \ldots, j_{n_a(t)}$ are the active jobs in MultiRAW at time t arranged in increasing order of arrival times, and $h(j_i) = \sum_{k=1}^{i} w(j_i)$. As mentioned in Section 4, $\Phi(t) = \gamma \sum_{i=1}^{n_a(t)} f(\frac{h(j_i)}{m}) \cdot x(j_i, t)$, where $\gamma = \frac{1}{\beta(1+\epsilon)}$ and $f(h) = h/Q(h)$.

Lemma 5. *At any time t without discrete events, $\frac{d\Phi}{dt} \le \frac{1-2\beta}{\beta} \cdot \max\{w_a(t), w_o(t)\} - \frac{1}{\beta^2}(1-\beta)\phi(t)$.*

PROOF. We consider $\frac{d\Phi}{dt}$ as a combined effect due to the action of MultiRAW and OFF. Let $\frac{d\Phi_a}{dt}$ and $\frac{d\Phi_o}{dt}$ be the rate of change of Φ due to MultiRAW and OFF, respectively. Then $\frac{d\Phi}{dt} = \frac{d\Phi_a}{dt} + \frac{d\Phi_o}{dt}$. Note that OFF is an offline algorithm on a single processor with maximum speed $m \cdot T$, which at any time t, follows the speed $m \cdot Q(\frac{w_o(t)}{m})$, where $w_o(t)$ is the total weight of active jobs in LLB$'$.

Consider the schedule of MultiRAW. For each active job j in MultiRAW, if $j \in L(t) \subseteq R(t)$, j is an unsaturated job, $x(j,t) > 0$ and $h(j) \ge (1-\beta)w_a(t)$. By Lemma 10, MultiRAW processes j at a rate $(1+\epsilon)\frac{w'(j,t)}{\beta w_a(t)} \cdot m \cdot Q(\frac{w_a(t)}{m})$. Hence, for each term $\gamma \cdot f(\frac{h(j)}{m}) \cdot x(j,t)$ where $j \in L(t)$, the term is decreasing at a rate at least

$$\begin{aligned} & \gamma \cdot \tfrac{h(j)/m}{Q(h(j)/m)} \cdot (1+\epsilon)\tfrac{w'(j,t)}{\beta w_a(t)} \cdot m \cdot Q(\tfrac{w_a(t)}{m}) \\ \ge\ & \gamma \cdot \tfrac{(1-\beta)w_a(t)/m}{Q((1-\beta)w_a(t)/m)} \cdot (1+\epsilon)\tfrac{w'(j,t)}{\beta w_a(t)} \cdot m \cdot Q(\tfrac{w_a(t)}{m}) \\ \ge\ & \gamma\tfrac{1-\beta}{\beta}(1+\epsilon)w'(j,t) = \tfrac{1-\beta}{\beta^2}w'(j,t) \end{aligned}$$

where the first inequality comes from that $f(h) = h/Q(h)$ is increasing with h, and the second inequality comes from that Q is increasing. For each job $j \notin L(t)$, the term $\gamma \cdot f(\frac{h(j)}{m}) \cdot x(j,t)$ is non-increasing. Hence, $\frac{d\Phi_a}{dt} \le -\frac{1-\beta}{\beta^2}\sum_{j \in L(t)} w'(j,t) = -\frac{1-\beta}{\beta^2}\phi(t)$.

The processing rate of an unsaturated job j in OFF is at most its speed on the single processor, which is $s_o = m \cdot Q(\frac{w_o(t)}{m})$. Thus, the worst case (in which $\frac{d\Phi_o}{dt}$ is the largest) occurs when OFF processes the job with maximum $h(j)$, which equals $w_a(t)$, using all its speed s_o. Then $\frac{d\Phi_o}{dt} \le \gamma \cdot f(\frac{w_a(t)}{m}) \cdot s_o = \gamma\frac{w_a(t)/m}{Q(w_a(t)/m)} \cdot m \cdot Q(w_o(t)/m)$. If $w_a(t) \ge w_o(t)$, $Q(w_a(t)/m) \ge Q(w_o(t)/m)$ and $\frac{d\Phi_o}{dt} \le \gamma w_a(t)$. Otherwise, $w_a(t) < w_o(t)$, since f is non-decreasing, $f(\frac{w_a(t)}{m}) \le f(\frac{w_o(t)}{m})$. Hence, $\frac{d\Phi_o}{dt} \le \gamma f(\frac{w_o(t)}{m})s_o = \gamma w_o(t)$. Combining these two cases, we obtain that $\frac{d\Phi_o}{dt} \le \gamma \cdot \max\{w_a(t), w_o(t)\}$. Finally, notice that $\beta = \frac{\epsilon}{2(1+\epsilon)}$ and $\frac{1}{1+\epsilon} = 1-2\beta$. Hence, $\gamma\max\{w_a(t), w_o(t)\} = \frac{1}{\beta(1+\epsilon)}\max\{w_a(t), w_o(t)\} = \frac{1-2\beta}{\beta}\max\{w_a(t), w_o(t)\}$ and the lemma follows. □

Appendix B: Offline Schedule Transformation

This appendix shows the following lemma on the performance of LLB$'$ that transforms OPT (which minimizes weighted flow plus energy on m processors) to an offline algorithm LLB$'$ on a single processor with maximum speed $m \cdot T$, which at any time t, follows the speed $m \cdot Q(\frac{w_b(t)}{m})$, where $w_b(t)$ is the total weight of active jobs in LLB$'$.

LEMMA 15. *The weighted flow of* LLB$'$ *is at most the weighted flow plus energy of* OPT.

Before proving Lemma 15, we first restate the algorithm LLB$'$ given in Section 4.

Algorithm LLB$'$. Consider any time t. Let $n_b(t)$ be the number of active jobs in LLB$'$. For any job j, let $p_b(j,t)$ and $p_o(j,t)$ be the work done on j up to time t by LLB$'$ and OPT, respectively. Furthermore, let $d(j,t) = p_o(j,t) - p_b(j,t)$. We say a job j is *lagging* at time t if $d(j,t) > 0$. Let $y(j)$ be the latest time when j has become lagging; if j is non-lagging, let $y(j) = t$. We denote the active jobs in LLB$'$ at time t as $j_1, j_2, \ldots, j_{n_b(t)}$, arranged in increasing order of $y(j_i)$, i.e., $y(j_1) \le y(j_2) \le \cdots \le y(j_{n_b(t)})$. Let ℓ be the number of lagging jobs ($0 \le \ell \le n_b(t)$). Let s_o be the total speed of the m processors in OPT at time t. Let $s_x \le s_o$ be the total speed OPT assigns to all jobs j with $d(j,t) = 0$. LLB$'$ sets its speed $s_b = m \cdot Q(\frac{w_b(t)}{m})$, and focuses on the job j_a defined to be j_ℓ if $\ell > 0$, and $j_{n_b(t)}$ otherwise. Details are as follows:

1. If OPT is not running any job j_i with $d(j_i,t) = 0$, then LLB$'$ runs j_a with speed s_b.
2. If OPT is processing some jobs j_i with $d(j_i,t) = 0$, but $s_x \le s_b$ (i.e., LLB$'$ has enough speed to prevent all of them from becoming lagging), then LLB$'$ follows OPT's speed on each of those jobs and runs j_a with the remaining speed $s_b - s_x$.
3. Otherwise ($s_b < s_x$), LLB$'$ follows OPT's speed for each of those jobs j_i, in descending order of the index i, until there is some job j which LLB$'$ cannot follow OPT's total speed on j. LLB$'$ then assigns all of its remaining speed to j.

Note that any job phase has a speed-up function Γ satisfying $\Gamma(y) = y$ for $y \in (0,1]$. Therefore, LLB$'$ using a single processor with maximum speed $m \cdot T$ can guarantee all its speed to be fully utilized to process the jobs and no speed would be wasted.

To prove Lemma 15, we exploit potential functions. Let F_b be the total weighted flow of LLB$'$, and let F_o and E_o be the total weighted flow and energy of OPT, respectively. For any time t, let $F_b(t)$ be the total weighted flow incurred up to time t by LLB$'$. Define $F_o(t)$ and $E_o(t)$ similarly. We derive a potential function $\Phi(t)$ that satisfies the following three conditions:

- *Boundary condition*: $\Phi = 0$ before the first job arrives and after all jobs are finished.
- *Discrete event condition*: Φ does not increase when a job arrives, or is completed by LLB$'$ or OPT, or when a lagging job is changed to non-lagging or vice versa.

- *Running condition*: At any other time t, $\frac{\mathrm{d}F_b(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le \frac{\mathrm{d}F_o(t)}{\mathrm{d}t} + \frac{\mathrm{d}E_o(t)}{\mathrm{d}t}$.

Integrating these conditions over time, we can conclude that $F_b \le F_o + E_o$; Lemma 15 follows.

Potential function $\Phi(t)$. Consider any time t. Recall that the active jobs in LLB$'$ are denoted as $j_1 \ldots j_{n_b(t)}$. Define the coefficient c_i of j_i to be $\sum_{k=1}^{i} w(j_k)$. The potential function $\Phi(t)$ is defined as follows.

$$\Phi(t) = \sum\nolimits_{i=1}^{n_b(t)} f(c_i) \cdot \max(d(j_i,t),0)$$

where $f(x) = P'(P^{-1}(\frac{x}{m}))$. Note that P' is the first derivative of P. Since P is convex, P' is non-decreasing, which together with that $P^{-1}(x)$ is non-decreasing, implies that $P'(P^{-1}(x))$ is also non-decreasing. Therefore, $f(x) = P'(P^{-1}(\frac{x}{m}))$ is a non-decreasing function of x.

The boundary condition is obvious. Now we check the discrete event condition. Recall that ℓ is the number of lagging jobs. When a job j arrives at time t, we have $d(j,t) = 0$, $y(j) = t$ and the coefficients of all lagging jobs of LLB$'$ remain the same, so Φ does not change. When OPT completes a job or LLB$'$ completes a non-lagging job, Φ does not change. When LLB$'$ completes a lagging job or a lagging job changes to non-lagging at time t, that job must be j_ℓ and $d(j_\ell,t) \le 0$. The coefficients of other lagging jobs stay the same, so Φ does not change. When one or more job(s) changes from non-lagging to lagging, those jobs must have the largest index among all lagging jobs and their $d(j,t)$'s are all 0. The coefficients of other lagging jobs stay the same, so Φ does not change.

It remains to check the running condition. Consider any time t without discrete events. Recall that s_b is the current speed of LLB$'$, and s_o is the current total processor speed of OPT. Let $w_\mathrm{o}(t)$ be the total weight of active jobs in OPT. Then, $\frac{\mathrm{d}F_b}{\mathrm{d}t} = w_b(t)$ and $\frac{\mathrm{d}F_o}{\mathrm{d}t} = w_\mathrm{o}(t)$. Since P is convex, the energy usage of the m processors in OPT is $\frac{\mathrm{d}E_o}{\mathrm{d}t} \ge m \cdot P(\frac{s_\mathrm{o}}{m})$. If all active jobs in LLB$'$ are non-lagging, i.e., $d(j_i,t) \le 0$ for all $1 \le i \le n_b(t)$, then these jobs are also active in OPT and hence $w_\mathrm{o}(t) \ge w_b(t)$. In this case, Φ remains zero and thus $\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} = 0$. Then the running condition follows easily since $\frac{\mathrm{d}F_b(t)}{\mathrm{d}t} = w_b(t) \le w_\mathrm{o}(t) \le \frac{\mathrm{d}F_o(t)}{\mathrm{d}t} + \frac{\mathrm{d}E_o(t)}{\mathrm{d}t}$. Henceforth, we assume at least one active job in LLB$'$ is lagging, i.e., $\ell > 0$.

We define a real number $\beta \le 1$ such that $\beta w_b(t) = c_\ell$, which is the total weight of lagging jobs. Then the rate of change of Φ can be bounded easily (Lemma 16). More interestingly, we can also show that at any time t, $Q(\beta w_b(t)) \le m \cdot T$ (Lemma 17).

LEMMA 16. $\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le f(\beta w_b(t)) \cdot (-s_b + s_\mathrm{o})$.

PROOF. We consider how Φ changes in an infinitesimal amount of time (from time t to $t + \mathrm{d}t$) where no job arrives, or completes, or changes from lagging to non-lagging or vice versa. Recall that s_x is OPT's total speed on jobs j_i with $d(j_i,t) = 0$. Then, OPT's total speed on jobs $j_{i'}$ with $d(j_{i'},t) \ne 0$ is $s_\mathrm{o} - s_\mathrm{x}$.

In the definition of LLB$'$, if Case 3 happens at t, some jobs including job j become lagging, which corresponds to a discrete event condition. Right after t, j become the new j_a and LLB$'$ will follow Case 1 or Case 2. Thus, it suffices to consider Cases 1 and 2 for the running condition.

Consider each job j_i with $d(j_i,t) = 0$. By Cases 1 and 2 in the definition of LLB$'$, either both LLB$'$ and OPT are not working on j_i, or they are running j_i with the same speed (totaling to s_x for all such j_i's). Therefore, $d(j_i,t)$ remains 0 and the processing of j_i does not affect the value of Φ.

Now consider other jobs j_i with $d(j_i,t) \ne 0$. Recall that LLB$'$ is running j_ℓ with the remaining speed $s_b - s_\mathrm{x}$. Since LLB$'$ is using a single processor with maximum speed $m \cdot T$, j_ℓ is also processed by LLB$'$ at a rate of $s_b - s_\mathrm{x}$. Consider the processing of OPT. The worst case that causes the largest increase in $\frac{\mathrm{d}\Phi}{\mathrm{d}t}$ is that OPT is also using all its remaining speed $s_\mathrm{o} - s_x$ to run j_ℓ. It is because any job j_k with $k > \ell$ is non-lagging and OPT cannot increase Φ by processing j_k. Also, OPT cannot increase Φ by processing a job that is not active in LLB$'$. In this case, $d(j_\ell,t)$ changes at a rate of at most $-(s_b - s_x) + (s_\mathrm{o} - s_x) = (-s_b + s_\mathrm{o})$. Therefore, Φ changes at rate at most $f(c_\ell) \cdot (-s_b + s_\mathrm{o}) = f(\beta w_b(t)) \cdot (-s_b + s_\mathrm{o})$. □

LEMMA 17. *At any time t, we have* $P^{-1}(\frac{\beta w_b(t)}{m}) \le T$.

PROOF. We will prove $\frac{\beta w_b(t)}{m} \le P(T)$; then the lemma follows by taking P^{-1} on both sides of the inequality. Consider the current time t. If $\frac{w_b(t)}{m} \le P(T)$, then $\frac{\beta w_b(t)}{m} \le \frac{w_b(t)}{m} \le P(T)$. If $\frac{w_b(t)}{m} > P(T)$, let t_0 be the last time before t where $\frac{w_b(t)}{m} \le P(T)$. Then the total weight of lagging jobs at t_0 is at most $m \cdot P(T)$. At any time after t_0, LLB$'$'s speed is $m \cdot T$. Since the total speed of OPT is also at most $m \cdot T$, by the definition of LLB$'$, for each job arrived after t_0, LLB$'$'s progress is at least OPT's progress. Hence, those new jobs cannot be lagging. Furthermore, Case 3 in the definition of LLB$'$ does not occur, and thus the total weight of lagging jobs cannot increase after t_0 and remains at most $m \cdot P(T)$ at time t. In other words, $\beta w_b(t) \le m \cdot P(T)$, i.e., $\frac{\beta w_b(t)}{m} \le P(T)$. □

We are now ready to prove the running condition, which together with the boundary and discrete event conditions, implies Lemma 15.

LEMMA 18. *At any time t without discrete events,* $\frac{\mathrm{d}F_b(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le \frac{\mathrm{d}F_o(t)}{\mathrm{d}t} + \frac{\mathrm{d}E_o(t)}{\mathrm{d}t}$.

PROOF. Recall that j_ℓ is the lagging job with the largest index, and $j_{\ell+1}, \ldots, j_{n_\mathrm{a}}$ are non-lagging and must also be active jobs in OPT. The total weight of these jobs is $w_b(t) - \beta w_b(t) = (1-\beta)w_b(t)$, so $w_\mathrm{o}(t) \ge (1-\beta)w_b(t)$. Also recall that $\frac{\mathrm{d}F_b(t)}{\mathrm{d}t} = w_b(t)$ and $\frac{\mathrm{d}F_o(t)}{\mathrm{d}t} + \frac{\mathrm{d}E_o(t)}{\mathrm{d}t} \ge w_\mathrm{o}(t) + m \cdot P(\frac{s_\mathrm{o}}{m}) \ge (1-\beta)w_b(t) + m \cdot P(\frac{s_\mathrm{o}}{m})$. We note a fact (given in [3]) that for any real $x \ge 0$, $P'(P^{-1}(x)) \cdot s_\mathrm{o} \le P^{-1}(x) \cdot P'(P^{-1}(x)) + P(s_\mathrm{o}) - x$. Lemma 16, together with this fact, gives

$$\begin{aligned}
&\tfrac{\mathrm{d}F_b(t)}{\mathrm{d}t} + \tfrac{\mathrm{d}\Phi(t)}{\mathrm{d}t}\\
&\le w_b(t) + P'(P^{-1}(\tfrac{\beta w_b(t)}{m})) \cdot (-s_b + s_\mathrm{o})\\
&\le w_b(t) + (-s_b)P'(P^{-1}(\tfrac{\beta w_b(t)}{m})) + m \cdot P'(P^{-1}(\tfrac{\beta w_b(t)}{m}))(\tfrac{s_\mathrm{o}}{m})\\
&\le w_b(t) + (-s_b)P'(P^{-1}(\tfrac{\beta w_b(t)}{m})) +\\
&\qquad m \cdot (P^{-1}(\frac{\beta w_b(t)}{m}) \cdot P'(P^{-1}(\frac{\beta w_b(t)}{m})) + P(\frac{s_\mathrm{o}}{m}) - \frac{\beta w_b(t)}{m})\\
&\le w_b(t) - \beta w_b(t) + m \cdot P(\tfrac{s_\mathrm{o}}{m}) +\\
&\qquad P'(P^{-1}(\frac{\beta w_b(t)}{m})) \cdot (-s_b + m \cdot P^{-1}(\frac{\beta w_b(t)}{m}))\ .
\end{aligned}$$

Since $s_b = m \cdot Q(\frac{w_b(t)}{m})$, where $Q(y) = \min(P^{-1}(y), T)$, we have $s_b = m \cdot \min(P^{-1}(\frac{w_b(t)}{m}), T)$. By Lemma 17, $s_b \ge m \cdot \min(P^{-1}(\frac{w_b(t)}{m}), P^{-1}(\frac{\beta w_b(t)}{m})) = m \cdot P^{-1}(\frac{\beta w_b(t)}{m})$. Thus, the last term of the above inequality is at most 0, and hence $\frac{\mathrm{d}F_b(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le (1-\beta)w_b(t) + m \cdot P(\frac{s_\mathrm{o}}{m}) \le \frac{\mathrm{d}F_o(t)}{\mathrm{d}t} + \frac{\mathrm{d}E_o(t)}{\mathrm{d}t}$. □