Primal Dual Gives Almost Optimal Energy Efficient Online Algorithms

Nikhil R. Devanur[*] Zhiyi Huang[†]

Abstract

We consider the problem of online scheduling of jobs on unrelated machines with dynamic speed scaling to minimize the sum of energy and weighted flow time. We give an algorithm with an almost optimal competitive ratio for arbitrary power functions. (No earlier results handled arbitrary power functions for minimizing flow time plus energy with unrelated machines.) For power functions of the form $f(s) = s^\alpha$ for some constant $\alpha > 1$, we get a competitive ratio of $O(\frac{\alpha}{\log \alpha})$, improving upon a previous competitive ratio of $O(\alpha^2)$ by Anand et al. [3], along with a matching lower bound of $\Omega(\frac{\alpha}{\log \alpha})$. Further, in the resource augmentation model, with a $1+\epsilon$ speed up, we give a $2(\frac{1}{\epsilon}+1)$ competitive algorithm, with essentially the same techniques, improving the bound of $1+O(\frac{1}{\epsilon^2})$ by Gupta et al. [15] and matching the bound of Anand et al. [3] for the special case of fixed speed unrelated machines. Unlike the previous results most of which used an amortized local competitiveness argument or dual fitting methods, we use a primal-dual method, which is useful not only to analyze the algorithms but also to design the algorithm itself.

1 Introduction

The design of online algorithms for scheduling problems has been an active area of research. Typically in such problems *jobs* arrive online, over time, and in order to complete a job it must be assigned a certain amount of processing, its processing *volume*. The algorithm has to schedule the jobs on one or more machines so as to complete them as soon as possible. A standard objective is a *weighted* sum of *flow times*; the flow time of a job is the duration of time between its release and completion. In the *unrelated* machines version of the problem, each job can have a different volume and a different weight for each machine. *Preemption/resumption* is allowed, but *migration* of a job from one machine to another is not.

Of late, an important consideration in such problems has been the *energy* consumption. A popular approach to model the energy consumption is via the *dynamic speed scaling* model: a machine can run at many different speeds, higher speeds process jobs faster but consume more energy. The rate of consumption of energy w.r.t. time is the *power* consumption which is given as a function of speed. Typical power functions are of the form s^α where s is the speed and α is some constant, commonly equal to 2 or 3. The objective now is to minimize the sum of energy consumed and weighted flow time. In this paper we show that a natural and principled primal-dual approach gives almost optimal competitive ratios for the most general version of the problem, with *unrelated* machines and *arbitrary* monotone non-decreasing and convex *power functions*.[1] (See Section 2 and Section 3 for formal problem definitions.) We summarize our contributions below.

For power functions $f(s) = s^\alpha$, we give an algorithm with competitive ratio $8\alpha/\log_2 \alpha$. We also give a corresponding lower bound of $\alpha/4\log_2 \alpha$. This improves upon a previous $O(\alpha^2)$ competitive ratio of Anand et al. [3]. We also show bounds for specific values of α, for $\alpha = 2$ and 3 we show competitive ratios of 4 and 5.581 respectively. *It is worth noting that these bounds improve upon the previous best bounds known for these values of α, even for a single machine* (which were 5.24 and 8 respectively for $\alpha = 2$ and 3, due to Bansal et al. [7]).[2] For an arbitrary power function f, we define a quantity Γ_f such that for the power function $f(s) = s^\alpha$, $\Gamma_f = \alpha$. Therefore Γ_f can be thought of as a generalization of the quantity α. We show analogous bounds for arbitrary power functions, an upper bound of $8\Gamma_f/\log_2 \Gamma_f$ and a lower bound of $\Gamma_f/4\log_2 \Gamma_f$. No previous bounds were known for arbitrary power functions.[3] These results are summarized in Table 1.

An alternate objective function often considered because it is sometimes easier to deal with, is the

[*]Microsoft Research, Redmond. Email: nikdev@microsoft.com.

[†]Stanford University. Email: hzhiyi@stanford.edu. This work was done while the author was a graduate student at University of Pennsylvania and an intern at Microsoft Research, Redmond.
Supported in part by an ONR MURI Grant N000140710907 and a Simons Graduate Fellowship for Theoretical Computer Science (No. 252128).

[1]This is w.l.o.g. as one can reduce problems with arbitrary power functions to those with non-decreasing and convex power functions (e.g., Section 3.1 of [6]).

[2] For the *unweighted* case, however, a competitive ratio of 2 is known, due to Andrew et al. [4].

[3]The power functions can differ by machine, and the competitive ratio is determined by the worst one.

fractional flow time. For the fractional flow time, imagine that a job is broken into infinitesimally small pieces and each piece has an independent flow time which is equal to the time between its own completion time and its release time. The fractional flow time is the average flow time of all the pieces put together. Our guarantees for the fractional flow time are essentially the same as those for the integral flow time and they are summarized in Table 2.

We also consider the *resource augmentation* model, where for the same given power, machines used by the algorithm run $1+\epsilon$ times faster than the machines used by the offline optimum. Our techniques (with minor modifications) extend to this model as well. We show a competitive ratio of $2(\frac{1}{\epsilon}+1)$ which improves upon the previous best known bound of $1+O(\frac{1}{\epsilon^2})$ by Gupta et al. [15]. As a special case when the power function is a 0-∞ step function, this bound matches the best known competitive ratio for minimizing the weighted flow time on fixed speed unrelated machines (obtained in Anand et al. [3] and Chadha et al. [11]). These results are summarized in Table 3.

Techniques and proof overview The most common and successful technique for analyzing online algorithms for such scheduling problems has been the *amortized local competitiveness* argument. The technique calls for a potential function that "stores" the excess cost incurred by the optimum solution and uses it as needed to pay for the algorithm's solution. More recently Anand et al. [3] used the dual fitting method to give several improved competitive ratios. We use a *primal dual* approach, which is a principled approach that is used to guide the design of the algorithm itself in addition to being a tool for the analysis. Our algorithms are based on a convex programming relaxation. Most of the work done is in understanding the structure of the convex program and the properties of the optimal primal and dual solutions; the algorithm and the analysis follow naturally after that. In other words, we derive the algorithm from the structure of the convex program.

Almost all the special cases of our problem, such as related machines, a single machine, unweighted flow time, etc. use the following speed scaling rule introduced by Albers and Fujiwara [2]: set the speed so that the power consumed is equal to the remaining weight (PERW). The power is the rate of energy consumed and the remaining weight can be thought of as the rate of accrual of the weighted flow time. What the PERW rule therefore ensures is that the energy consumed and the weighted flow time are both equal, which is convenient for the analysis. At first glance, it may also appear that PERW is the greedy choice, i.e., it is the optimal choice if no more jobs are released. A more careful analysis shows this to be false, that the optimal speed to set, if no more jobs are released, is s so that $f^*(f'(s))$ equals the remaining weight.[4] We first analyze the natural greedy algorithm that follows from this speed scaling rule coupled with a greedy job assignment policy: a job is assigned to the machine for which the increase in the energy plus flow time is the smallest. This gives an $O(\alpha)$ competitive ratio, which already beats the previous bound of $O(\alpha^2)$ which uses the greedy job assignment policy along with the PERW speed scaling rule.

Setting the speed so that no other job arrives in the future is really a conservative approach. The algorithm should anticipate the arrival of some jobs in the future and run faster. Such an aggressive algorithm faces a trade-off: the improvement obtained when there are more jobs in the future versus the degradation when there aren't. The algorithm should hedge against both the cases and balance the competitive ratio. A systematic way to do this is to set the speed s so that $f^*(f'(s/c))$ equals remaining weight, for some constant c, obtain a competitive ratio as a function of c and then set c to minimize the competitive ratio. For instance, for the power function $f(s) = s^\alpha$, the best choice of c we have turns out to be $1+\frac{\ln\alpha-1}{\alpha}$, giving a competitive ratio of $\frac{8\alpha}{\log_2\alpha}$. Moreover this framework allows us to also analyze the PERW rule for power functions $f(s) = s^\alpha$, which corresponds to a choice of $c = (\alpha-1)^{1/\alpha}$. We show that the competitive ratio with the PERW speed scaling rule is still $O(\frac{\alpha}{\log\alpha})$, thus giving some justification for this rule as well.

We start our analysis by considering the objective of weighted fractional flow time plus energy (Section 2). We consider a convex programming relaxation and its dual using Fenchel conjugates. We analyze the simple case of scheduling a single job on a single machine and derive the speed scaling rule $f^*(f'(s)) =$ remaining (fractional) weight as optimal. We then consider many jobs on a single machine, all released at time 0, and show that the same speed scaling rule is optimal, along with the job selection rule of highest density first (HDF, density = weight/volume). We characterize the optimal dual solution for the same and show several structural properties of the optimal primal and dual solutions. Finally we consider the unrelated machines case, which calls for a job assignment rule (which assigns jobs to machines). At any time, given the job assignments, the algorithm uses the optimal

[4] f^* is the *Fenchel conjugate* of f, defined as $f^*(\mu) := \sup_x\{\mu x - f(x)\}$, and $f^*(f'(s))$ has the following geometric interpretation: if you draw a tangent to f at s, then this is the length of the y-intercept of the tangent.

Table 1: Competitive ratios for minimizing integral weighted flow time plus energy (Section 3)

	Best known			This paper		
	$\alpha = 2$	$\alpha = 3$	General	$\alpha = 2$	$\alpha = 3$	General
Single machine, $f(s) = s^\alpha$	5.24 [7]	8 [7]	$O(\frac{\alpha}{\log_2 \alpha})$ [6, 7][c]	4	5.581	$O(\frac{\alpha}{\log_2 \alpha})$[a] (Thm. 3.2)
Unrelated machines, $f(s) = s^\alpha$			$O(\alpha^2)$ [3]			$\Theta(\frac{\alpha}{\log_2 \alpha})$[a] (Thm. 3.2, 5.1)
Unrelated machines, any $f(s)$			(new)			$\Theta\left(\frac{\Gamma_f}{\log_2 \Gamma_f}\right)$[ab] (Thm. 3.2, 5.1)

Table 2: Competitive ratios for minimizing fractional weighted flow time plus energy (Section 2)

	Best known	This paper			
	General	$1 < \alpha < 2$	$\alpha = 2$	$\alpha = 3$	General
Single machine, arbitrary $f(s)$	2 [6]	α	2	2.791	$O(\frac{\alpha}{\log_2 \alpha})$
Unrelated machines, $f(s) = s^\alpha$	$O(\alpha)$ [3]	2α	4	5.581	$\Theta(\frac{\alpha}{\log_2 \alpha})$[a] (Thm. 2.3, 5.1)
Unrelated machines, any $f(s)$	(new)				$\Theta\left(\frac{\Gamma_f}{\log_2 \Gamma_f}\right)$[a] (Thm. 2.3, 5.1)

Table 3: Competitive ratios for minimizing fractional/integral weighted flow time plus energy (any power function $f(s)$) with resource augmentation ($(1+\epsilon)$-speed) (Section 4)

	Best known	This paper
Integral flow time plus energy, arbitrary power function	$1 + O(\frac{1}{\epsilon^2})$ [15]	$2 + \frac{2}{\epsilon}$ (Thm. 4.1)
Fractional flow time plus energy, arbitrary power function	$1 + \frac{5}{\epsilon}$[15]	
Integral flow time, fixed speed	$2 + \frac{2}{\epsilon}$ [3]	$2 + \frac{2}{\epsilon}$ (Thm. 4.2)
Fractional flow time, fixed speed	$2 + \frac{2}{\epsilon}$ [11]	

[a] Specifically, we show an upper bound of $\frac{8\Gamma_f}{\log_2 \Gamma_f}$ ($\frac{8\alpha}{\log_2 \alpha}$ for $f(s) = s^\alpha$) and a lower bound of $\frac{\Gamma_f}{4 \log_2 \Gamma_f}$ ($\frac{\alpha}{4 \log_2 \alpha}$ for $f(s) = s^\alpha$) on the competitive ratios.

[b] Γ_f is defined to be $\max_s f^*(f'(s))/f(s) + 1$.

[c] To achieve this competitive ratio, one need to combine the techniques in [6, 7]. E.g., see the discussions by Anand et al. [3] for more details.

schedule for each machine assuming no future jobs arrive and the corresponding dual. The job assignment rule follows from the complementary slackness conditions using these duals.[5] The competitive ratio of $O(\alpha)$ follows from a simple local charging of the primal to the dual cost, along with some of the structural properties established earlier. We then consider a systematically aggressive speed scaling rule, $f^*(f'(s/c))$ = remaining weight for some constant c (Section 2.5). With the rest of the algorithm/proof more or less identical, we derive a competitive ratio as a function of c. On optimizing, we get a competitive ratio of $O(\alpha/\log\alpha)$.

Typically algorithms designed for the fractional flow time also work for the integral flow time with some loss in the competitive ratio. However, our analysis for the fractional flow time also goes through for the integral flow time (with small modifications) without any loss in the competitive ratio! Almost the same proof structure works for the integral case and we outline what minor modifications are required (Section 3). Once again essentially the same proof goes through for the resource augmentation model as well (Section 4). Finally we show a simple example with 2 machines that gives almost tight lower bounds (Section 5).

Related work We summarize here a selection of the most relevant related work. The fact that energy costs are a substantial part of the overall cost in data centers (see Barroso [9]) motivates energy considerations in scheduling problems. Early work on energy efficient algorithms was for the case of a single machine. Albers and Fujiwara [2] introduced the objective of energy plus flow time in the dynamic speed scaling model, and the PERW speed scaling rule. Generalizations to weighted flow time followed [7, 5, 17] with the current best competitive ratios given by Bansal et al. [6] and Andrew et al. [4]. For other variants in the dynamic speed scaling model, and other models such as the power down model and the importance of energy efficient algorithms, see the survey by Albers [1].

Motivated by the design of architectures with heterogenous cores/processors, Gupta et al. [15] considered the case of related machines with different power functions. (See the references therein for more reasons to consider heterogenous machines.) All these algorithms use the PERW speed scaling rule and use potential functions with amortized local competitiveness arguments. Anand et al. [3] used a dual fitting based argument to generalize this to unrelated machines and the fixed speed case in the resource augmentation model (improving upon a previous algorithm by Chadha et al. [11]). The major difference between dual fitting and primal dual is that dual fitting is only used as an analysis tool for a given algorithm while primal dual guides the design of the algorithm itself. Also in recent work Gupta et al. [14] showed that the natural extensions of several well known algorithms that work for homogeneous machines fail for heterogeneous machines, thus justifying the use of "non-standard" algorithms of Gupta et al. and Anand et al. [15, 3]. As summarized in Table 1-3, our work unifies and improves upon several of these results, most prominently [7, 6, 3, 15, 11].

Buchbinder and Naor [10] established the primal-dual approach for packing and covering problems, unifying several previous potential function based analysis. Gupta et al. [16] gave a primal-dual algorithm for a general class of scheduling problems with cost functions of the form $f(s) = s^\alpha$. A dual (and equivalent) problem of online concave matching was considered by Devanur and Jain [13] who also used the primal-dual approach to give optimal competitive ratios for arbitrary cost functions. Using either of these results for the problems we consider only gives a competitive ratio of α^α which is significantly far from the ratios we obtain.

Recent related work In concurrent and independent work, Nguyen [18] shows an $O(\alpha/\log\alpha)$-competitive algorithm for minimizing flow time plus energy on unrelated machines when the power function is $f(s) = s^\alpha$. The analysis uses dual fitting based on a non-convex program and Lagrangian duality.

2 Weighted fractional flow-time plus energy

In this section we consider the objective of fractional flow time plus energy and obtain competitive algorithms for it. Suppose that we are given a power function, $f : \mathbb{R}_+ \mapsto \mathbb{R}_+$, which is monotonically non-decreasing, convex and is 0 at 0. Further, we assume that f is also differentiable. For ease of presentation we assume that all the machines have the same power function although all of our results go through easily with different power functions for different machines. First, we define the **offline** version of the problem.

Input: A set of jobs J and a set of machines M. For each job $j \in J$ its release time $r_j \in \mathbb{R}_+$. For each job $j \in J$ and each machine $i \in M$, the *volume* v_{ij} and the *weight* w_{ij} of job j if scheduled on machine i. Let the *density* of job j on machine i be $\rho_{ij} = w_{ij}/v_{ij}$.

[5] This assignment rule is almost the same as the greedy assignment rule, but is slightly different. We state the job assignment rule derived from the complementary slackness conditions in the algorithm since that seems more principled. In any case, the analysis remains the same for both job assignment rules.

Output: An assignment of each job to a single machine (*no migration*). For each machine i and time $t \in \mathbb{R}_+$, the job scheduled at time t on machine i, denoted by $j_i(t)$ and the *speed* at which the machine is run, denoted by s_{it}.

Constraints: Job j must be scheduled only after its release time. It must recieve a total amount of v_{ij} units of computation if it is assigned to machine i, i.e., $\int_{t\in[r_j,+\infty]:j_i(t)=j} s_{it}\, dt = v_{ij}$.

Objectives: The objective has two components, energy and fractional flow-time. Recall that f is the power function, which gives the power consumption as a function of the speed. The energy consumed by machine i is therefore

$$E_i = \int_0^\infty f(s_{it})dt \ .$$

The fractional flow-time is an aggregated measure of the waiting time of a job. Suppose job j is scheduled on machine i. Let $\hat{v}_j(t)$ be the remaining volume of job j at time t, i.e.,

$$\hat{v}_j(t) = v_{ij} - \int_{t'\in[r_j,t]:j_i(t')=j} s_i(t')dt' \ .$$

The fractional flow-time of job j is then defined to be

$$F_j := \frac{1}{v_{ij}} \int_{r_j}^\infty \hat{v}_j dt \ .$$

The objective is to minimize the total energy consumed by all the machines and a weighted sum of the flow-times of all the jobs:

$$\sum_i \left(E_i + \sum_{j:j\to i} w_{ij} F_j \right) \ .$$

In the **online** version of the problem the details of job j are given only at time r_j. The algorithm has to make decisions at time t without knowing anything about the jobs released in the future.

2.1 Convex programming relaxation and the dual The algorithms we design are based on a convex programming relaxation of the problem and its dual, which are shown in Figure 1. The dual convex program is obtained using Fenchel duality. The Fenchel conjugate of f is the function $f^*(\mu) := \sup_x \{\mu x - f(x)\}$. (See Appendix A or [12] for a detailed explanation.)

The variables s_{ijt} denote the speed at which job j is scheduled on machine i at time t. $s_{it} = \sum_j s_{ijt}$ is the total speed of machine i at time t. For each job j, constraint (2.1) enforces that the scheduling must complete job j. (We will not state trivial constraints such as $s_{ijt} \geq 0$ throughout the paper.) In the objective function, the first summation corresponds to the fractional flow-time: $s_{ijt}dt$ units of job j is processed between t and $t+dt$, all of which waited for a duration of $t-r_j$ resulting in $(t-r_j)\frac{s_{ijt}}{v_{ij}}dt$ amount of fractional flow-time. The second summation is the total energy consumed.

The third summation is required because the convex program allows a job to be split among many machines and even have different parts run in parallel. Without the third term, the convex program fails to provide a good lower bound on the cost of the optimal solution. We show that the optimum of the convex program with an additional factor of 2 is a lower bound on OPT, the optimum offline solution to the problem. We note this in the following theorem.

$$\begin{aligned}
(\mathrm{P_{frac}}) \quad & \text{minimize } \sum_{i,j} \rho_{ij} \int_{r_j}^\infty (t-r_j) s_{ijt} dt \\
& \quad + \sum_i \int_0^\infty f(s_{it})dt \\
& \quad + \sum_{i,j} \int_{r_j}^\infty \frac{s_{ijt}}{w_{ij}} \left(\int_0^{w_{ij}} (f^*)^{-1}(w)dw \right) dt \\
\forall i,t: \quad & s_{it} = \sum_{j:r_j\le t} s_{ijt} \\
(2.1) \quad \forall j: \quad & \sum_i \int_{r_j}^\infty \frac{s_{ijt}}{v_{ij}} dt \geq 1
\end{aligned}$$

$$\begin{aligned}
(\mathrm{D_{frac}}) \quad & \text{maximize } \sum_j \alpha_j - \sum_i \int_0^\infty f^*(\beta_{it})dt \\
\forall i,j,t\geq r_j: \quad & \frac{\alpha_j}{v_{ij}} \leq \rho_{ij}(t-r_j) + \beta_{it} \\
& \quad + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw
\end{aligned}$$

Figure 1: Convex Programming relaxation of minimizing fractional flow time plus energy.

THEOREM 2.1. *The optimum value of the convex crogram* ($\mathrm{P_{frac}}$) *is at most* 2OPT.

Proof. Consider an instance with only one job released at time 0 and a large number of machines. The optimal solution to the convex program schedules the job simultaneously on all the machines and the total cost w.r.t. the first two terms will tend to zero as the number of machines tends to infinity. The optimal algorithm has to schedule the job on a single machine and hence pays a fixed non-zero cost. The third term fixes this problem: consider a modified instance where we have multiple copies of each machine (as many as the number of jobs); the cost of the optimal solution to this instance is only lower. In this modified instance, w.l.o.g., no two jobs are ever scheduled on the same machine. It can be shown (Lemma 2.2) that if job j is scheduled on a copy of machine i all by itself, then the optimal cost (energy + flow-time) due to job j is $\frac{v_{ij}}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw$. Now, still allowing a job to be split among different machines, an $\int_{r_j}^\infty \frac{s_{ijt}}{v_{ij}} dt$ fraction of job j is scheduled on machine i.

Thus $\sum_i \int_{r_j}^{\infty} \frac{s_{ijt}}{w_{ij}} \left(\int_0^{w_{ij}} (f^*)^{-1}(w)dw\right) dt$ is a lower bound on the cost of scheduling job j.

The algorithm heavily uses the structure of the optimal solutions to the primal and the dual programs. We explain this structure in stages. There is a natural decomposition of the problem itself: at the highest level is the decision to allocate a job to one of the machines. Given these choices, the rest of the problem decomposes into a separate one for each machine. For each machine, given the set of jobs that have to be scheduled on it and the volumes and densities, there is the problem of picking the job to schedule and the speed to set at any time, in order to minimize the total energy and flow-time on that machine. Further, given the choice of the job to schedule on a machine, there is an even simpler problem of setting the speed. We start with the simplest problem of all, given just a single job and a single machine, what is the optimal speed schedule that minimizes the total energy and fractional flow-time.[6]

2.2 Optimal scheduling for single job

We obtain a simpler convex program and its dual for the problem of scheduling a single job on a single machine. We drop the third term in the objective since that deals with non-integral assignment of jobs to machines. Since there is only one job in this subsection, we assume w.l.o.g. that $r_j = 0$.

$$\begin{aligned} \text{minimize} \quad & \int_0^\infty \rho_{ij} t s_{ijt} dt + \int_0^\infty f(s_{it}) dt \\ \forall t \ : \quad & s_{it} = s_{ijt} \\ & \int_0^\infty \frac{s_{ijt}}{v_j} dt \geq 1 \end{aligned} \tag{2.2}$$

$$\begin{aligned} \text{maximize} \quad & \alpha_j - \int_0^\infty f^*(\beta_{it}) dt \\ \forall t \ : \quad & \frac{\alpha_j}{v_{ij}} \leq \rho_{ij} t + \beta_{it} \end{aligned}$$

Recall that the conjugate function f^* is defined as $f^*(\beta) := \sup_s\{\beta s - f(s)\}$. The function f^* is also convex and monotonically non-decreasing. If f is strictly convex, then so is f^*. The most important property we use about f^* is the notion of a *complementary pair* . β and s are said to be a complementary pair if any one of the following conditions hold. (It can be shown that if one of them holds, then so do the others.)

1. $f'(s) = \beta$;
2. $(f^*)'(\beta) = s$;
3. $f(s) + f^*(\beta) = s\beta$.

[6] A characterization of the optimal scheduling can be found, e.g., in [8], but we believe it is illustrative to rederive the optimal scheduling using our primal dual approach.

The optimal solutions to these programs are characterized by the (generalized) complementary slackness or KKT conditions. These are:

1. $\forall t, \ s_{ijt} > 0 \Rightarrow \alpha_j = v_{ij}(\rho_{ij} t + \beta_{it})$;
2. $\alpha_j > 0 \Rightarrow \int_0^\infty \frac{s_{ijt}}{v_j} dt = 1$;
3. β_{it} and s_{it} are a complementary pair for all t.

The first condition implies that for the entire duration that the machine is running (with non-zero speed), the quantity $\rho_{ij}t+\beta_{it}$ remains the same, since it must always equal α_j. In other words, β_{it} must linearly decrease with time, at the rate of ρ_{ij}, i.e., $\frac{d\beta_{it}}{dt} = -\rho_{ij}$.

The main result in this section is that the optimum solution has a closed form expression where s_{it} and β_{it} are set as a function of the remaining weight of the job at time t, which we denote by $\hat{w}_{it}$. (More generally, $\hat{w}_{it}$ will denote the total remaining weight of all the jobs on machine i.) Also the remaining volume at time t is denoted $\hat{v}_{it}$.

Lemma 2.1. *The optimum solution to the convex program Eqn. (2.2) is such that $f^*(f'(s_{it})) = f^*(\beta_{it}) = \hat{w}_{it}$, and $\alpha_j = v_{ij} f^{*-1}(w_{ij})$.*

Proof. Since β_{it} and s_{it} form a complementary pair, we have that $\frac{df^*(\beta_{it})}{d\beta_{it}} = s_{it}$. We start by multiplying the LHS by $\frac{d\beta_{it}}{dt}$ and the RHS by $-\rho_{ij}$, which is valid since we showed these are equal. This gives

$$\frac{df^*(\beta_{it})}{d\beta_{it}} \frac{d\beta_{it}}{dt} = -s_{it}\rho_{ij} \quad .$$

Therefore

$$\frac{df^*(\beta_{it})}{dt} = \rho_{ij} \frac{d\hat{v}_{it}}{dt} = \frac{d\hat{w}_{it}}{dt} \quad .$$

When $\hat{w}_{it} = 0$, then $s_{ijt} = 0$, $f(s_{ijt}) = 0$ and therefore $f^*(\beta_{it}) = 0$ (by the third property of complementary pairs). Therefore at any time $f^*(\beta_{it}) = \hat{w}_{it}$. The rest of the assertions in the lemma follow immediately.

Remark 2.1. *By $f^*(\beta_{it}) = \hat{w}_{it}$ and that β_{it} and s_{it} are a complementary pair, we get a closed form of $s_{it} = (\alpha - 1)^{-1/\alpha} \hat{w}_{it}^{1/\alpha}$ when $f(s) = s^\alpha$. Contrast this with the PERW rule for which $s_{it} = \hat{w}_{it}^{1/\alpha}$.*

We also give a closed form for the value of the optimum. This form justifies the inclusion of the third term in the objective for the convex program (P_{frac}). We will also use this lemma later to analyze how the cost of the optimum solution changes as we add new jobs.

Lemma 2.2. *The cost of the optimal solution is* $\frac{1}{\rho_{ij}}\int_0^{w_{ij}}(f^*)^{-1}(w)dw$.

Proof. Recall that the total weighted flow-time is equal to $\int_0^\infty \hat{w}_{it}dt$ and the total energy is equal to $\int_0^\infty f(s_{it})dt$. From Lemma 2.1, $f^*(\beta_{it}) = \hat{w}_{it}$. Using this and the properties of complementary pairs, we get the following sequence of equalities.

$$(\hat{w}_{it} + f(s_{it}))\,dt = (f^*(\beta_{it}) + f(s_{it}))\,dt = \beta_{it}s_{it}dt\ .$$

Further, by the definition of s_{it} and β_{it}, the above equals

$$-\beta_{it}d\hat{v}_{it} = -\frac{1}{\rho_{ij}}\beta_{it}d\hat{w}_{it} = -\frac{1}{\rho_{ij}}f^{*-1}(\hat{w}_{it})d\hat{w}_{it}\ .$$

The lemma follows from observing that as t goes from 0 to ∞, $\hat{w}_{it}$ goes from w_{ij} to 0.

2.3 Optimal scheduling for a single machine We now consider the next stage where there are multiple jobs to be scheduled on a single machine, and the corresponding convex programs. We will continue to assume that $r_j = 0$ for all jobs j.

$$\text{(2.3)}\qquad \begin{array}{rl} \text{minimize} & \sum_j \int_{r_j}^\infty \rho_{ij}ts_{ijt}dt + \int_0^\infty f(s_{it})dt \\ \forall t\ : & s_{it} = \sum_j s_{ijt} \\ \forall j\ : & \int_0^\infty \frac{s_{ijt}}{v_{ij}}dt \geq 1 \end{array}$$

$$\begin{array}{rl} \text{maximize} & \sum_j \alpha_j - \int_0^\infty f^*(\beta_{it})dt \\ \forall t, j\ : & \frac{\alpha_j}{v_{ij}} \leq \rho_{ij}t + \beta_{it} \end{array}$$

The complementary slackness conditions for these pair of programs are more or less as before. To begin with, $s_{ijt} > 0 \Rightarrow \frac{\alpha_j}{v_{ij}} = \rho_{ij}t + \beta_{it}$. As before, this implies that β_{it} decreases at rate ρ_{ij} whenever job j is scheduled, but the main new issue is the choice of jobs to schedule. The above complementary slackness condition implies that job j must be scheduled when the term $\rho_{ij}t+\beta_{it}$ attains its minimum. The first part, $\rho_{ij}t$, always increases at rate ρ_{ij}, while the second part, β_{it} decreases at rate $\rho_{ij(t)}$ where $j(t)$ is the job scheduled at time t. So if $\rho_{ij} < \rho_{ij(t)}$ then $\rho_{ij}t+\beta_{it}$ is decreasing and vice-versa, if $\rho_{ij} > \rho_{ij(t)}$ then $\rho_{ij}t+\beta_{it}$ is increasing. This implies that the "highest density first" (HDF) rule is optimal, i.e., schedule the jobs in the decreasing order of the density. For any j, $\rho_{ij}t + \beta_{it}$ first decreases when higher density jobs are scheduled, then remains constant as job j is scheduled and then increases as lower density jobs are scheduled.

Given the choice of jobs scheduled, the choice of speed is very similar to the single-job case. We state the following generalization of Lemma 2.1 without proof, since it either follows from the discussion above or is very similar to the proof of Lemma 2.1.

Lemma 2.3. *The optimum solution to the convex program* (2.3) *is such that*

1. *Jobs are scheduled in the decreasing order of density.*
2. $f^*(f'(s_{it})) = f^*(\beta_{it}) = \hat{w}_{it}$.
3. $\alpha_j = w_{ij}t^* + v_{ij}f^{*-1}(\hat{w}_{it^*})$ *where* t^* *is the first time job* j *is scheduled.*

Unlike the single-job case, we no longer have a closed form expression for the optimal cost. Instead, we consider the marginal increase in the optimal cost due to a single job, and show the following generalization of Lemma 2.2.

Lemma 2.4. *The increase in the cost of the optimal solution on introducing a new job* j *is at most*

$$w_{ij}t^* + \frac{1}{\rho_{ij}}\int_0^{w_{ij}} f^{*-1}(\hat{w}_{it^*} + w)dw$$

where t^* *is the first time job* j *is scheduled and* $\hat{w}_{it^*}$ *is the remaining weight at time* t^* *in the original instance, without job* j.

Proof. It suffices to show that even if we use a sub-optimal speed-scaling after introducing the new job j, the increase in the cost is only the amount as stated in the lemma. In particular consider the sub-optimal scheduling in which we keep the schedule till time t^* unchanged, and then start to schedule job j and use the optimal speed scaling. The entire job j waits till time t^* contributing $w_{ij}t^*$ to the total flow-time. For the rest of the increase, consider introducing an infinitesimal weight dw at time t. This causes the following infinitesimal increase in the flow-time and energy: $dF = \hat{w}_{it}dt$ and $dE = f(s_{it})dt$. The rest of the proof is along the same lines as that of Lemma 2.2 and is omitted here.

Finally, we note a couple of simple observations that follow almost immediately from Lemma 2.3. Let $\Gamma_f = \max_s \frac{f^*(f'(s))}{f(s)} + 1$.

Lemma 2.5. $\int_0^\infty f^*(\beta_{it})dt$ *equals the total flow-time.*

Lemma 2.6. *The total flow-time is at most* $\Gamma_f - 1$ *times the total energy consumed.*

Proof. The total energy used is $\int_0^\infty f(s_{it})dt$. The total flow-time is $\int_0^\infty \hat{w}_{it}dt$. Recall that by Lemma 2.3, s_{it} is chosen such that $\hat{w}_{it} = f^*(f'(s_{it}))$. So the ratio between the fractional flow time and the energy is $\int_0^\infty f^*(f'(s_{it}))dt$ divided by $\int_0^\infty f(s_{it})dt$, which is at most $\max_s \frac{f^*(f'(s))}{f(s)}$.

2.4 Conservative greedy algorithm In this section, we analyze a primal-dual algorithm which we call conservative-greedy. The basic idea is that given the choice of job assignments to machines, the algorithm schedules the jobs as if no other jobs will be released in the future. That is, it schedules the jobs as per the optimal schedule for the current set of jobs, as detailed in the previous section. The choice of job assignments to machines is done via a natural primal-dual method, the one dictated by the complementary slackness conditions.

Concretely, at any point, given the jobs already released and their assignment to machines, the algorithm picks the optimal scheduling on each machine, assuming no future jobs are released. This also gives dual solutions, in particular the variables β_{it} for all i and t in the future. When a new job j is released, its assignment to a machine is naturally driven by the following dual constraints and the corresponding complementary slackness conditions. For all i, t,

$$\frac{\alpha_j}{v_{ij}} \le \rho_{ij}(t - r_j) + \beta_{it} + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw.$$

For a given machine i, we saw earlier that the right hand side (RHS) is minimized (over all t) at t_i^* where t_i^* would be the first time job j is scheduled on i given the HDF rule. That still holds true since the third term above is independent of t. Now we need to minimize over all i as well and the algorithm does exactly this. It assigns job j to the machine i that minimizes the RHS of the inequality above with $t = t_i^*$ multiplied by v_{ij}. It sets the dual α_j so that the corresponding constraint is tight. It then updates the schedule and β_{it}'s for machine i. Note that as we add more jobs, the β_{it}'s can only increase, thus preserving dual feasibility. The entire algorithm is summarized in Figure 2.

We will show that, surprisingly, such a conservative approach already achieves a meaningful competitive ratio for arbitrary power functions and a near optimal competitive ratio for polynomial power functions. Formally, we will show the following theorem in this section.

THEOREM 2.2. *The fractional conservative greedy algorithm is $2\Gamma_f$-competitive for minimizing weighted fractional flow time plus energy.*

The above competitive ratio might be unbounded if the function is highly skewed. Theorem 2.2 does not contradict known lower bounds for fixed-speed online scheduling, which is a special case when the power function is a 0-∞ step function. For nice power functions such as polynomial power functions, the above theorem gives meaningful competitive ratios. In particular, the fractional conservative greedy algorithm is 2α-competitive for minimizing weighted fractional flow time plus energy with $f(s) = s^\alpha$.

In the remaining of this section, we will present the primal-dual analysis of Theorem 2.2. It is easy to see that the algorithm constructs feasible primal and dual solutions and the ratio is obtained by relating the cost of the primal to that of the dual. In fact, for every job released, we relate the increase in the cost of the primal to the increase in the cost of the dual.

LEMMA 2.7. *When job j is released, the increase in the total cost of the algorithm is at most α_j.*

Proof. Suppose job j is assigned to machine i and will be scheduled at t^* (dropping subscript i) according to the HDF rule on the current set of jobs assigned to i. Then

$$\alpha_j = w_{ij}(t^* - r_j) + v_{ij}\beta_{it^*} + \frac{1}{\rho_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw \ .$$

The increase in the total cost of the algorithm is only the increase in that for machine i. From Lemma 2.4 this is at most

$$w_{ij}(t^* - r_j) + \frac{1}{\rho_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w + \hat{w}_{it^*})dw \ .$$

Comparing the two, it suffices to show that

$$\text{(2.4)} \quad \beta_{it^*} + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw \ge \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w + \hat{w}_{it^*})dw \ .$$

By Lemma 2.3, we have $\beta_{it^*} = (f^*)^{-1}(\hat{w}_{it^*})$. Further, $(f^*)^{-1}$ is concave because f^* is convex. So

$$(f^*)^{-1}(\hat{w}_{it^*}) + (f^*)^{-1}(w) \ge (f^*)^{-1}(w + \hat{w}_{it^*})$$

for all $0 \le w \le w_{ij}$. Integrate this inequality for w from 0 to w_{ij} and we get Eqn. (2.4) as desired.

Proof. [Theorem 2.2] Consider the release of a new job j and suppose it is assigned to machine i. The change in the dual cost, ΔD, equals α_j plus the change in the contribution of β_{it}'s, i.e., $\Delta D = \alpha_j - \Delta\left(\int_0^\infty f^*(\beta_{it})dt\right)$. Let the change in the total energy and the total flow-time of the algorithm be ΔE and ΔF respectively. From Lemma 2.7, $\alpha_j \ge \Delta E + \Delta F$. Earlier in Lemma 2.5 we showed that the total flow-time is always equal to $\int f^*(\beta_{it})dt$. Thus the same holds for the difference. $\Delta\left(\int f^*(\beta_{it})dt\right) = \Delta F$. From the three inequalities above, the change in the dual cost is at least the change in the energy cost of the algorithm. Since this holds for

Fractional conservative greedy algorithm

Speed scaling: Choose speed s_{it} s.t. $f^*(f'(s_{it}))$ equals the fractional remaining weight on machine i. Set duals $\beta_{it} = f'(s_{it})$, also for future times based on the planned schedule currently.

Job selection: Schedule the job with the highest density (HDF).

Job assignment: Assign job j to machine i that minimizes $\rho_{ij}(t_i^* - r_j) + \beta_{it_i^*} + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw$ where t_i^* would be the first time job j is scheduled on i given the HDF rule. Set α_j so that the corresponding constraint is tight. Update the β_{it}'s for machine i.

Figure 2: The conservative greedy online scheduling algorithm for minimizing fractional flow time plus energy with arbitrary power functions

every change and both the dual cost and the energy cost of the algorithm are zero to begin with, it follows that the final dual cost is at least the final energy cost of the algorithm, i.e., $\Delta D \geq \Delta E \Rightarrow D \geq E$.

Further, by Lemma 2.6, the total flow-time and the energy are within a factor of $\Gamma_f - 1$. Even though that was for a single machine with no future jobs, it is easy to see that the same holds true for the conservative-greedy algorithm as well. Therefore

$$F \leq (\Gamma_f - 1)E \ .$$

The total cost of the algorithm, ALG can now be bounded in terms of the energy cost alone, which is bound by the dual as above, i.e.,

$$\text{ALG} = E + F \leq \Gamma_f E \ .$$

Also by Theorem 2.1 the dual is a lower bound on 2OPT, i.e.,

$$D \leq 2\text{OPT} \ .$$

Putting it all together, we get

$$\text{ALG} \leq 2\Gamma_f \text{OPT} \ .$$

An alternate algorithm with essentially the same analysis is the following: assign job j to machine i for which the increase in the total cost is the minimum. The dual α_j must however be set as we do currently, so there might be a disconnect between which machine the job is assigned to and which machine dictates the dual solution. The analysis still is pretty much the same however.

2.5 Aggressive greedy algorithm In the conservative greedy algorithm, the speed is scaled to the conservative extreme as the speed is optimal assuming no future jobs arrive. However, in an online instance there might be jobs in the future, some of which will be effectively delayed by the current job. Therefore, a good online algorithm should take this into account when choosing the speed. In this section, we consider a family of algorithms with the aggressiveness in terms of speed scaling parameterized by any constant $C \geq 1$, as given in Figure 3.

The following property of the β_{it}'s implies that our choice of β_{it}'s can be derived from the primal dual analysis and ensures dual feasibility.

Lemma 2.8. *For any time t at which no new jobs are released, β_{it} linearly decrease with time, at the rate of ρ_{ij}, where j is the job being processed on i at time t, i.e., $\frac{d\beta_{it}}{dt} = -\rho_{ij}$.*

Proof. [Sketch] Note that the claimed equation is satisfied by the conservative greedy algorithm. Further, the C-aggressive greedy algorithm is running at exactly C times the speed of the conservative greedy algorithm given the same remaining weight of jobs, and the β_{it}'s are set to be $\frac{1}{C}$ fraction of those in the conservative greedy given the same remaining weight of jobs. Putting together these observations and our definition of β_{it} we can easily verify the lemma.

We will show that by choosing the optimal aggressiveness we can improve the competitive ratio by a logarithmic factor. Concretely, we show the following.

Theorem 2.3. *The fractional C-aggressive greedy algorithm is $2\Gamma_{f,C}$-competitive for minimizing weighted fractional flow time plus energy with power function f, where*

$$\Gamma_{f,C} = \frac{C(C^{\Gamma_f} + \Gamma_f - 1)}{\Gamma_f C - \Gamma_f + 1} \ .$$

(Recall that $\Gamma_f = \max_s \frac{f^*(f'(s))}{f(s)} + 1$.)

Proof of Theorem 2.3 In the rest of this section, we will present of proof of Theorem 2.3. Given Lemma 2.8, it is easy to check that the way we are setting the primal and dual variables guarantees feasibility. The competitive ratio comes from analyzing the ratio between the

Fractional C-aggressive greedy algorithm

Speed scaling: Choose speed s_{it} s.t. $f^*(f'(\frac{s_{it}}{C}))$ equals the total remaining weight on machine i. Set duals $\beta_{it} = \frac{1}{C}(f^*)^{-1}(\hat{w}_{it})$ s.t. $f^*(C\beta_{it})$ equals the total remaining weight on machine i, also for future times based on the planned schedule currently.

Job selection: Schedule the job with highest density (HDF).

Job assignment: Assign job j to machine i that minimizes $\rho_{ij}(t_i^* - r_j) + \beta_{it_i^*} + \frac{1}{w_{ij}}\int_0^{w_{ij}}(f^*)^{-1}(w)dw$ where t_i^* would be the first time job j is scheduled on i given the HDF rule. Set α_j so that the corresponding constraint is tight. Update the β_{it}'s for machine i.

Figure 3: The aggressive greedy online scheduling algorithm for minimizing weighted fractional flow time plus energy with arbitrary power functions

incremental costs of the algorithm and the dual due to the arrival of new jobs.

We will first develop a few lemmas that are needed in the analysis. We start by showing that the parameter Γ_f for arbitrary function f plays a similar role as the degree of polynomial functions as the following lemma holds. (Recall $\Gamma_f = \alpha$ for $f(s) = s^\alpha$.)

Lemma 2.9. *For any power function f, any $s > 0$ and any $C > 1$, we have*

$$f(Cs) \le C^{\Gamma_f} f(s) \ .$$

Proof. Note that for any $s_1 > 0$, we have that

$$\frac{f'(s_1)s_1}{f(s_1)} = \frac{f^*(f'(s_1)) + f(s_1)}{f(s_1)} = \frac{f^*(f'(s_1))}{f(s_1)} + 1 \le \Gamma_f \ ,$$

where the first equality is because s_1 and $f'(s_1)$ are a complementary pair and the inequality is by definition of Γ_f. So we have that

$$\begin{aligned} \ln(f(Cs)) - \ln(f(s)) &= \textstyle\int_1^C d\ln(f(xs)) \\ &= \textstyle\int_1^C \frac{f'(xs)s}{f(xs)}dx \\ &= \textstyle\int_1^C \frac{f'(xs)xs}{f(xs)}\frac{1}{x}dx \\ &\le \textstyle\int_1^C \Gamma_f \frac{1}{x}dx = \Gamma_f \ln(C) \ . \end{aligned}$$

The lemma follows.

Alternative dual objective: For the convenience of analysis, we will consider an alternative dual objective,

$$\max \quad \sum_j \alpha_j - \sum_i \int_0^\infty \frac{1}{C} f^*(C\beta_{it})dt \ ,$$

subject to the same constraints. We let $\widehat{D}$ denote the value of the above dual objective. By the convexity of f^*, we have the following lemma.

Lemma 2.10. *For any values of the dual variables, we have $\widehat{D} \le D$.*

Alternative power function: We will consider the total cost of our algorithm w.r.t. the power function $\hat{f}(s) = C^{\Gamma_f} f(\frac{s}{C})$. We let $\widehat{E}$ denote the energy cost of the algorithm w.r.t. this power function, and let $\widehat{\text{ALG}} = F + \widehat{E}$ denote the total cost of the algorithm w.r.t. this power function, noting that the flow time is unaffected. By Lemma 2.9, we have

Lemma 2.11. *For any instance, we have $\widehat{E} \ge E$ and $\widehat{\text{ALG}} \ge \text{ALG}$.*

By Lemma 2.10 and Lemma 2.11, it suffices to bound the ratio between the increase in the total cost of the algorithm w.r.t. power function $\hat{f}$ when a new job arrives and the increase in the alternative dual objective, denoted as $\Delta\widehat{\text{ALG}}$ and $\Delta\widehat{D}$ respectively.

Next, suppose a new job j arrives at time t and is assigned to machine i. We will account for the incremental costs of the algorithm and the dual by relating them to the incremental cost in an imaginary instance in which we are using the conservative greedy algorithm. We will call the current instance I and the imaginary instance I'.

More precisely, let there be a single machine in I' that is identical to machine i in I. For each incomplete job in I at time t (before the arrival of job j), we put an identical job with the same remaining volume in I' at time 0. Then, we will consider also releasing job j in I' at time 0. By our construction of I' and the fact that the C-aggressive greedy algorithm is running exactly C-times faster than the conservative one, there is a one-to-one mapping between the timeline after t in I and the timeline in I' as specified in the next lemma, whose proof is straightforward and omitted.

Lemma 2.12. *For any time $t' \ge t$, the remaining weight of each job in I at time t' is the same as that in I' at time $C(t' - t)$, both before and after the release of job j.*

Let $\Delta F'$ and $\Delta E'$ denote the increase in weighted fractional flow time and energy respectively in I' conditioned on running the conservative greedy both before and after releasing job j. Recall that in the conservative greedy algorithm, we have $\Delta F' = (\Gamma_f - 1)\Delta E'$.

We let ΔF and $\Delta \widehat{E}$ denote the increase in weighted fractional flow time and energy (w.r.t. $\hat{f}$) in I due to the arrival of job j.

The next lemma accounts for the the increase in flow time due to the arrival of job j as a fraction of $\Delta E'$. The proof is rather straightforward from Lemma 2.12 so we omit it.

Lemma 2.13. *$\Delta F = \frac{1}{C}\Delta F' = \frac{1}{C}(\Gamma_f - 1)\Delta E'$.*

The next lemma bound the increase in energy due to the arrival of job j by a fraction of $\Delta E'$.

Lemma 2.14. *$\Delta \widehat{E} = C^{\Gamma_f - 1}\Delta E'$.*

Proof. For any $t' \geq t$, suppose the C-aggressive greedy algorithm runs at speed s at time t'. Then, the energy cost (w.r.t. $\hat{f}$) in I from time t' to $t' + dt'$ is $\hat{f}(s)dt' = C^{\Gamma_f} f(\frac{s}{C})dt'$. Further, by Lemma 2.12 and that the C-aggressive greedy algorithm is running C-times faster than the conservative one, the conservative greedy algorithm runs at speed $\frac{s}{C}$ at time $C(t' - t)$ in I'. So the energy cost of from time $C(t' - t)$ to $C(t' - t) + Cdt'$ in I' is $f(\frac{s}{C})Cdt'$, a $C^{-\Gamma_f + 1}$ fraction of the corresponding energy cost in I. Integrating for all $t' \geq t$, we get that the energy cost (w.r.t. $\hat{f}$) in I after time t is exactly $C^{\Gamma_f - 1}$ times the total energy cost in I'. As this relation holds both before and after the release of job j, the lemma follows.

The next lemma establish the relation between the incremental cost of the alternative dual due to β_{it}'s and $\Delta E'$.

Lemma 2.15. *$\Delta \int_0^\infty \frac{1}{C} f^*(C\beta_{it})dt = \frac{1}{C^2}(\Gamma_f - 1)E'$.*

Proof. [Sketch] Recall that $\beta_{it} = \frac{1}{C}(f^*)^{-1}(\hat{w}_{it})$. By the convexity of f^*, we have $\frac{1}{C}f^*(C\beta_{it}) = \frac{1}{C}\hat{w}_{it}$. So we have that $\int_0^\infty \frac{1}{C}f^*(C\beta_{it})dt$ equals $\frac{1}{C}$ times the weighted fractional flow time of instance I. Hence, we have

$$\Delta \int_0^\infty \frac{1}{C} f^*(C\beta_{it})dt = \frac{1}{C}\Delta F = \frac{1}{C^2}\Delta F' \ .$$

The lemma then follows from $\Delta F' = (\Gamma_f - 1)\Delta E'$.

The next lemma follows from the fact that the C-aggressive greedy is running at exactly C time the speed of conservative greedy and our choice of β_{it}.

Lemma 2.16. *$\alpha_j \geq \frac{1}{C}(\Delta F' + \Delta E') = \frac{1}{C}\Gamma_f \Delta E'$.*

Proof. [Sketch] Consider the time t^* at which the new job j would be inserted according to HDF w.r.t. the original instance (without job j). Let t' denote the time at which j would be inserted in the imaginary instance. By our choice of speed scaling, we have $(t^* - t) = \frac{1}{C}t'$. Moreover, $\beta_{it^*} = \frac{1}{C}(f^*)^{-1}(\hat{w}_{it})$. So by our choice of α_j and an analysis similar to Lemma 2.7, we get that α_j is at least $\frac{1}{C}$ time the total increase in weighted fractional flow time plus energy in I', i.e., $\alpha_j \geq \frac{1}{C}(\Delta E' + \Delta F')$.

Finally, we are ready to derive the competitive ratio of the C-aggressive greedy algorithm.

Proof. [Theorem 2.3] Putting together Lemma 2.13 to Lemma 2.16, we have that the incremental costs of the algorithm (w.r.t. $\hat{f}$) is

$$\Delta \widehat{\mathrm{ALG}} = \Delta F + \Delta \widehat{E} = \left(\frac{1}{C}(\Gamma_f - 1) + C^{\Gamma_f - 1}\right)\Delta E'$$

and the incremental costs of the alternative dual is

$$\begin{aligned} \Delta \widehat{D} &= \alpha_j - \Delta \frac{1}{C}\int_0^\infty f^*(C\beta_{it})dt \\ &\geq \left(\frac{1}{C}\Gamma_f - \frac{1}{C^2}(\Gamma_f - 1)\right)\Delta E' \ . \end{aligned}$$

Simplifying the ratio between $\Delta \widehat{\mathrm{ALG}}$ and $\Delta \widehat{D}$ from the above equations proves Theorem 2.3.

Optimal choice of aggressiveness By optimizing our choice of C for $1 < \Gamma_f \leq 2$, $\Gamma_f = 3$, and asymptotically optimizing it for general Γ_f, we get the following corollary. We also show its (asymptotic) optimality in Section 5 with an almost matching lower bound.

Corollary 2.1. *The fractional C-aggressive greedy algorithm is:*

- *$2\Gamma_f$-competitive for $1 < \Gamma_f \leq 2$, with $C = 1$;*
- *5.581-competitive for $\Gamma_f = 3$, with $C \approx 1.168$;*
- *$\frac{8\Gamma_f}{\log_2 \Gamma_f}$-competitive for general Γ_f, with $C = 1 + \frac{\ln \Gamma_f - 1}{\Gamma_f}$.*

Proof. The competitive ratios for $1 < \Gamma_f \leq 2$ and $\Gamma_f = 3$ are easy to verify. So we omit the tedious calculation here.

Next, consider the asymptotical bound for general Γ_f. With our choice of $C = 1 + \frac{\ln \Gamma_f - 1}{\Gamma_f}$, the denominator of $\Gamma_{f,C}$ is

$$\Gamma_f C - \Gamma_f + 1 = \ln \Gamma_f = \log_2 \Gamma_f / \log_2(e) \ .$$

On the other hand, note that

$$C = 1 + \frac{\ln \Gamma_f - 1}{\Gamma_f} = 1 + \frac{\ln(1 + (\Gamma_f - 1)) - 1}{\Gamma_f} \le 1 + \frac{\Gamma_f - 2}{\Gamma_f}$$

and

$$C^{\Gamma_f} = \left(1 + \frac{\ln \Gamma_f - 1}{\Gamma_f}\right)^{\Gamma_f} < e^{\ln \Gamma_f - 1} = \frac{1}{e}\Gamma_f \quad .$$

So putting together, we get that the numerator of $\Gamma_{f,C}$ is

$$\begin{aligned} C(C^{\Gamma_f} + \Gamma_f - 1) &< \left(1 + \frac{\Gamma_f - 2}{\Gamma_f}\right)\left(\frac{1}{e}\Gamma_f + \Gamma_f - 1\right) \\ &= \left(2 - \frac{2}{\Gamma_f}\right)\left(\frac{1}{e} + 1 - \frac{1}{\Gamma_f}\right)\Gamma_f \\ &\le \left(\frac{2}{e} + 2\right)\Gamma_f \quad . \end{aligned}$$

Finally, by the above discussion and that $(\frac{2}{e} + 2)\log_2(e) < 4$ and Theorem 2.3 we prove the claimed asymptotic competitive ratio for arbitrary Γ_f.

In particular, for polynomial power functions $f(s) = s^\alpha$, we have the following:

Corollary 2.2. *The fractional C-aggressive greedy algorithm is:*

- 2α*-competitive for* $1 < \alpha \le 2$*, with* $C = 1$*;*
- 5.581*-competitive for* $\alpha = 3$*, with* $C \approx 1.168$*;*
- $\frac{8\alpha}{\log_2 \alpha}$*-competitive for general* α*, with* $C = 1 + \frac{\ln \alpha - 1}{\alpha}$.

We remark that with $f(s) = s^\alpha$ the speed scaling of PERW also falls into the framework of C-aggressive greedy algorithm. Further, its choice of $C^\alpha = (\alpha - 1)$ is also asymptotically optimal when $\alpha \ge 2$. (We omit this calculation in this paper.) So our result can be viewed as a formal justification of the optimality of PERW. Similar remark applies to the integral case.

We also note that for minimizing fractional flow time plus energy, we can drop the third term in the primal objective and drop a factor of 2 in the competitive ratio (e.g., Lemma 2.6). As a result, we can get an α-competitive algorithm for $1 < \alpha \le 2$, and a 2.791-competitive algorithm for $\alpha = 3$, hence the claimed ratios in Table 2. We omit the details in this paper.

3 Weighted integral flow-time plus energy

In this section, we will discuss the problem of online scheduling for minimizing weighted (integral) flow-time plus energy. The problem for weighted integral flow time has the same input, output, and constraints as the fractional flow time version. The only difference is the objective. Next let us formally define the weighted integral flow time of an instance given a schedule. Suppose job j is completed at time c_j, then the weighted integral flow time is

$$\sum_i \sum_{j:j\to i} w_{ij}(c_j - r_j).$$

An equivalent formula for the same is as follows. Let A_t denote the set of jobs such that have been released before or at time t but have not been completed according to the schedule till time t, i.e.,

$$r_j \le t \quad \text{and} \quad \int_{t\in[r_j,\infty]:j_i(t)=j} s_{it}dt < v_{ij} \quad .$$

The weighted integral flow time is equal to

$$\int_0^\infty \sum_i \sum_{j\in A_t:j\to i} w_{ij}dt \quad .$$

So the main difference is that when a job is partially completed, the *entire* weight of the job will contribute to the weighted integral flow time, while only the *incomplete fraction* will contribute to the weighted fractional flow time.

Convex programming relaxation and the dual Similar to the fractional, we will use the primal-dual analysis via following convex program for the problem of minimizing integral flow time plus energy and consider its dual program:

$$\begin{aligned} (\mathrm{P_{int}}) \quad \text{minimize} \quad & \textstyle\sum_i \sum_j \int_{r_j}^\infty \rho_{ij}(t - r_j)s_{ijt}dt \\ & + \textstyle\int_0^\infty f(s_{it})dt \\ & + \textstyle\sum_i \sum_j \int_{r_j}^\infty (f^*)^{-1}(w_{ij})s_{ijt}dt \end{aligned}$$

$$(3.5) \quad \forall i, t: \quad \textstyle\sum_{j:r_j\le t} s_{ijt} = s_{it}$$

$$(3.6) \quad \forall j: \quad \textstyle\sum_i \int_{r_j}^\infty \frac{s_{ijt}}{v_{ij}} \ge 1$$

$$(\mathrm{D_{int}}) \quad \text{maximize} \quad \textstyle\sum_j \alpha_j - \sum_i \int_0^\infty f^*(\beta_{it})dt$$

$$(3.7) \quad \forall i, j, t \ge r_j: \quad \frac{\alpha_j}{v_{ij}} \le \rho_{ij}(t - r_j) + \beta_{it} + (f^*)^{-1}(w_{ij})$$

Here we use the same notation as in the fractional case so we will omit the detailed explanations of the

Integral conservative greedy algorithm

Speed scaling: Choose speed s_{it} s.t. $f^*(f'(s_{it}))$ equals the integral remaining weight on machine i. Set duals $\beta_{it} = f'(s_{it})$ s.t. $f^*(\beta_{it})$ equals the integral remaining weight on machine i, also for future times based on the planned schedule currently.

Job selection and job assignment: Upon arrival of a new job j, assign it to a machine i and insert it into the processing queue of i such that we minimize

$$\rho_{ij}(t_i^* - r_j) + \beta_{it_i^*} + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w)dw \ ,$$

where t_i^* would be the completion time of the predecessor of job j in the queue. Set α_j so that the corresponding constraint is tight. Update the β_{it}'s for machine i.

Figure 4: The conservative greedy online scheduling algorithm for minimizing weighted integral flow time plus energy with arbitrary power functions

convex programs. The only change is the third term in the primal program (and the corresponding part in the dual). This is because conditioned on being allocated to machine i, the optimal cost for job j in a single-job instance w.r.t. integral flow time plus energy is $v_{ij}(f^*)^{-1}(w_{ij})$. Hence, the share of the optimal single-job cost for the $\frac{s_{ijt}}{v_{ij}}dt$ fraction of job j is processed on machine i from t to $t + dt$ is $(f^*)^{-1}(w_{ij})s_{ijt}dt$.

Algorithms Similar to the fractional case, we will consider the conservative greedy algorithm, which will use the optimal speed scaling assuming there are no future jobs, and a more general family of C-aggressive greedy algorithms. The main difference comparing to the fractional case is the job selection rule on a single machine is no longer HDF. Instead, the algorithms will combine the job assignment rule job selection rule by maintaining a processing queue for each machine. The machines will process the jobs in their queues in order. When a new job arrives, the algorithm will insert the new job to an position in one of the processing queue according to the dual variables. The formal descriptions of the algorithms are presented in Figure 4 and Figure 5.

The integral conservative/aggressive greedy algorithms obtain the following competitive ratio for general power functions and polynomial power functions respectively.

Theorem 3.1. *The integral conservative greedy algorithm is $2\Gamma_f$-competitive for minimizing weighted integral flow time plus energy, where recall that $\Gamma_f = \max_s \frac{f^*(f'(s))}{f(s)} + 1$.*

Corollary 3.1. *The integral conservative greedy algorithm is 2α-competitive for power function $f(s) = s^\alpha$ for minimizing weighted integral flow time plus energy.*

Theorem 3.2. *The integral C-aggressive greedy algorithm is $2\Gamma_{f,C}$-competitive for minimizing weighted integral flow time plus energy with power function f, where*

$$\Gamma_{f,C} = \frac{C(C^{\Gamma_f} + \Gamma_f - 1)}{\Gamma_f C - \Gamma_f + 1} \ .$$

By asymptotically optimizing our choice of C, we get the following corollaries, whose optimality will be presented in Section 5. The proofs are identical to the integral case and hence omitted.

Corollary 3.2. *The integral C-aggressive greedy algorithm is*

- *$2\Gamma_f$-competitive for $1 < \Gamma_f \le 2$, with $C = 1$;*
- *5.581-competitive for $\Gamma_f = 3$, with $C \approx 1.168$;*
- *$\frac{8\Gamma_f}{\log_2 \Gamma_f}$-competitive for general Γ_f, with $C = 1 + \frac{\ln \Gamma_f - 1}{\Gamma_f}$.*

Corollary 3.3. *For minimizing weighted integral flow time plus energy w.r.t. power function $f(s) = s^\alpha$, the fractional C-aggressive greedy algorithm is*

- *2α-competitive for $1 < \alpha \le 2$, with $C = 1$;*
- *5.581-competitive for $\alpha = 3$, with $C \approx 1.168$;*
- *$\frac{8\alpha}{\log_2 \alpha}$-competitive for general α, with $C = 1 + \frac{\ln \alpha - 1}{\alpha}$.*

Competitive ratio The primal dual analysis of the integral case is almost identical to the fractional case. So here we will only sketch the analysis of the conservative algorithm and explain the main differences between the integral case and the fractional case. Extending the

Integral C-aggresive greedy algorithm

Speed scaling: Choose speed s_{it} s.t. $f^*(f'(\frac{s_{it}}{C}))$ equals the integral remaining weight on machine i. Set duals $\beta_{it} = \frac{1}{C} f'(\frac{s_{it}}{C})$ s.t. $f^*(C\beta_{it})$ equals the integral remaining weight on machine i, also for future times based on the planned schedule currently.

Job selection and job assignment: Upon arrival of a new job j, assign it to a machine i and insert it into the processing queue of i such that we minimize

$$\rho_{ij}(t_i^* - r_j) + \beta_{it_i^*} + \frac{1}{w_{ij}} \int_0^{w_{ij}} (f^*)^{-1}(w) dw \ ,$$

where t_i^* would be the completion time of the predecessor of job j in the queue. Set α_j so that the corresponding constraint is tight. Update the β_{it}'s for machine i.

Figure 5: The C-aggressive greedy online scheduling algorithm for minimizing weighted integral flow time plus energy with arbitrary power function

analysis from conservative to aggressive algorithms is fairly straightforward using techniques we introduce in Section 2.5. So the details will be omitted.

Next, we will show that α_j is an upper bound on the increase in flow time plus energy due the job j. This is the main technical component of the analysis of the competitive ratio. The claim follows easily from the next lemma, whose proof is similar to Lemma 2.7 and will be sketched below.

Lemma 3.1. *Suppose j' is a job in the queue of machine i whose scheduled completion time is $t_{ij'}$ before the arrival of job j. Then, the increase in total costs if we insert job j to the queue of machine i right after j' is at most*

$$v_{ij}\left(\rho_{ij}(t_{ij'} - r_j) + \beta_{it_{ij'}} + (f^*)^{-1}(w_{ij})\right)$$

Proof. [Sketch] Note that the algorithm uses the optimal speed scaling, assuming no future jobs. To bound the increase in flow time plus energy when we insert job j to the queue of machine i right after j', it suffices to bound the increase in flow time plus energy when we use a sub-optimal speed scaling. In particular, we will let the jobs scheduled before j use the same speed as before the arrival of j, and use the optimal speed scaling after that. In this case, the increase in flow time plus energy will be the flow time due to job j waiting until time $t_{ij'}$, i.e., $w_{ij}(t_{ij'} - r_j)$, plus the increase in flow time (of job j and jobs scheduled after j) plus energy due to processing job j, i.e., $v_{ij}(f^*)^{-1}(W(t_{ij'}) + w_{ij})$ where $W(t_{ij'})$ is the integral remaining weight of jobs on machine i at time $t_{ij'}$. By the concavity of $(f^*)^{-1}$, increase in flow time is at most

$$v_{ij}\left((f^*)^{-1}(W(t_{ij'})) + (f^*)^{-1}(w_{ij})\right) \ .$$

The lemma then follows by $f^*(\beta_{it_{ij'}}) = W(t_{ij'})$.

Note that β_{it} remains the same while machine i is processing the same job. So the minimal value of $v_{ij}\left(\rho_{ij}(t - r_j) + \beta_{it} + (f^*)^{-1}(w_{ij})\right)$ must be achieved in the completion time $t_{ij'}$ of one of the jobs on i. As a simple corollary of Lemma 3.1, we have:

Corollary 3.4. *The value of α_j is at least the increase in flow time plus energy due to job j.*

Now we are ready to analyze the competitive ratio of the integral conservative greedy algorithm.

Proof. [Theorem 3.1] First, note that it follows from our definition of the primal program $\mathrm{P}_{\mathrm{int}}$ that its optimal objective is at most twice that of the optimal flow time plus energy by any (offline) algorithm. Further, by our choice of β_{it}'s, the contribution of $\sum_i \int_0^\infty f^*(\beta_{it})dt$ is the weighted integral flow time of the algorithm. So combining with Corollary 3.4 we get that upon the arrival of new jobs, the increase in the dual objective is at least the increase in energy of the algorithm. Finally, via the same argument as in the fractional case we have that the weighted integral flow time of the algorithm is at most Γ_f times the energy. So the theorem follows.

4 Resource augmentation

Further, as a simple application of our primal dual approach, we manage to give simpler proofs for either matching or improved competitive ratios for several online scheduling problems with resource augmentation. In the resource augmentation setting, we will compare to a weaker offline benchmark in the sense that given the same energy, the offline algorithm can only run at $(1+\epsilon)^{-1}$ fraction of the speed of the online algorithm.

Theorem 4.1. *The fractional/integral conservative greedy algorithm is $(1+\epsilon)$-speed and $2\left(\frac{1}{\epsilon}+1\right)$-*

competitive for minimizing weighted fractional/integral flow time plus energy with arbitrary power functions.

As a simple corollary of the above theorem when the power functions are step functions, we have the following theorem for minimizing fractional/integral flow-time (fixed speed) with resource augmentation.

THEOREM 4.2. *The fractional/integral conservative greedy algorithm is $(1+\epsilon)$-speed and $2\left(\frac{1}{\epsilon}+1\right)$-competitive for for minimizing weighted fractional/integral flow time on fixed-speed machines.*

Primal dual analysis with resource augmentation We only need to slightly modify the primal dual analysis in Section 2 and Section 3 in order to prove Theorem 4.1. Here, we will sketch the proof for the minimizing weighted integral flow time. The analysis for weighted fractional flow time is almost identical and omitted.

First, we will need to change the objectives of the primal and dual programs to capture a weaker offline offline benchmark whose speed is only a $(1+\epsilon)^{-1}$ fraction of that of the online algorithm using the same energy. In particular, we will consider the following modified primal and dual convex programs:

$$\begin{aligned}\text{minimize} \quad & \textstyle\sum_i \sum_j \int_{r_j}^{\infty} \rho_{ij}(t-r_j) s_{ijt} dt \\ & + \textstyle\int_0^{\infty} f((1+\epsilon)s_{it}) dt \\ & + \textstyle\sum_i \sum_j \int_{r_j}^{\infty} (f^*)^{-1}(w_{ij}) s_{ijt} dt \\ \text{subject to} \quad & (3.5), (3.6)\end{aligned}$$

$$\begin{aligned}\text{maximize} \quad & \textstyle\sum_j \alpha_j - \sum_i \int_0^{\infty} f^*((1+\epsilon)^{-1}\beta_{it}) dt \\ \text{subject to} \quad & (3.7)\end{aligned}$$

We will use the same rule as in Section 3 for setting the primal and dual variables. Since the constraints remain the same, primal and dual feasibilities will be satisfied. Next, let us analyze the competitive ratio. Recall that by Corollary 3.4 we have that α_j is at least the increase in weighted flow time plus energy of the algorithm due to job j. Further, by our choice of β_{it} and the convexity of $f^*(\cdot)$ we have that

$$\begin{aligned} & \sum_i \int_0^{\infty} f^*((1+\epsilon)^{-1}\beta_{it}) dt \\ \leq \; & (1+\epsilon)^{-1} \sum_i \int_0^{\infty} f^*(\beta_{it}) dt \\ = \; & (1+\epsilon)^{-1} \sum_i \int_0^{\infty} W_i^*(t) dt \end{aligned}$$

is at most a $(1+\epsilon)^{-1}$ fraction of the weighted flow time of the algorithm. So the increase in the dual objective due to job j is at least the increase in energy plus a $1-(1+\epsilon)^{-1} = \frac{\epsilon}{1+\epsilon}$ fraction of the increase in flow time of the algorithm. Finally, recall that the optimal primal objective is at most twice the optimal weighted flow time plus energy of any (offline) algorithm. So we get that the integral conservative algorithm with $(1+\epsilon)$-speed-up is $2\left(\frac{1}{\epsilon}+1\right)$-competitive for minimizing weighted flow time plus energy.

5 Almost tight lower bounds

In this section, we complement our algorithmic results in Section 2 and Section 3 by providing asymptotically tight lower bounds for minimizing weighted fractional/integral flow-time plus energy with arbitrary power functions. Formally, we prove the following:

THEOREM 5.1. *Suppose there exists $\Gamma \geq 1$ and $s > 0$ such that for any $s' \in [s, s + \frac{\log_2 \Gamma}{\Gamma} s]$, we have $\frac{f^*(f'(s'))}{f(s')} + 1 \geq \Gamma$. Then, any online algorithm for scheduling jobs on unrelated machines with speed scaling to minimize weighted fractional/integral flow-time plus energy must have a competitive ratio of at least $\frac{1}{4}\frac{\Gamma}{\log_2 \Gamma}$.*

In particular, we have the following asymptotically tight lower bound for polynomial power functions as a corollary of Theorem 5.1.

COROLLARY 5.1. *Any online algorithm for scheduling jobs on unrelated machines with speed scaling to minimize weighted fractional/integral flow-time plus energy w.r.t. polynomial power function $f(s) = s^{\alpha}$ $(\alpha \geq 2)$ must have a competitive ratio of at least $\frac{1}{4}\frac{\alpha}{\log_2 \alpha}$.*

Proof. [Theorem 5.1] Note that a lower bound for fractional flow-time will also imply the same asymptotically tight lower bound for integral flow-time plus energy, because we can view each job in the fractional instance as continuously many infinitesimally small jobs with the same density in the integral instance. So in this proof, we only need to present the lower bound for fractional flow-time.

We construct a randomized instance consisting of two machines. Consider two types of jobs, both of which have density ρ. The value of ρ will be determined later. The first type consists of only 1 job of size 1 that arrives at time 0. This job can be processed by both machines. The second type consists of a sequence of jobs of total volume Γ that comes at rate s, i.e., from t to $t+dt$ we have $s \cdot dt$ volume of such jobs arriving, from time 0^+ (after the release of the type-1 job) to $\frac{\Gamma}{s}$. The type-2 jobs can only be processed by one of the machines, which is randomly chosen when we construct the instance (but it will be the same machine for all type-2 jobs).

Now we will show that even if the online algorithm knows the instance, but not the random coin flip, it must admit a competitive ratio at least $\Omega(\frac{\Gamma}{\log_2 \Gamma})$. By Yao's minimax principle [19], it suffices to consider deterministic algorithm A and lower bound its competitive ratio.

Let us first upper bound the cost of the offline optimal. The offline optimal could have processed all type-2 jobs on the feasible machine with a fixed speed s, and processed the type-1 job on the other machine also at a fixed speed s. The energy cost will be $\frac{\Gamma}{s} f(s)$ in total. The weighted fractional flow time will be $\frac{\rho}{2s}$ because the type-1 job incurs a weighted fractional flow time of $\frac{\rho}{2s}$ while the type-2 jobs are completed immediately and hence have 0 weighted fractional flow time. We will choose the density ρ to balance the two types of costs, i.e., $\frac{\Gamma}{s} f(s) = \frac{\rho}{2s}$. So the total cost is $\frac{\rho}{s}$.

Now consider the deterministic online algorithm. At time 0 the algorithm needs to decide which machine the type-1 job is assigned to. So with probability $\frac{1}{2}$ the algorithm will make a mistake and assign the type-1 job to the only machine that can process the type-2 jobs. We will show that in this case the cost of the online algorithm will be at least $\Omega(\frac{\Gamma}{\log_2 \Gamma} \frac{\rho}{s})$. Choose $t^* > 0$ such that

$$f\left(s + \frac{1}{2t^*}\right) = \Gamma f(s) \quad . \tag{5.8}$$

We claim that the following lemma about t^* holds.

LEMMA 5.1. $t^* \geq \frac{1}{2s} \frac{\Gamma}{\log_2 \Gamma}$.

We will need the following lemma about f, whose proof is similar to that of Lemma 2.9 and hence omitted.

LEMMA 5.2. *For any $C \in [1, 1 + \frac{\log_2 \Gamma}{\Gamma}]$, we have that*

$$f(Cs) \geq C^{\Gamma} f(s) \quad .$$

Proof. [Lemma 5.1] First note that Lemma 5.2 implies

$$f\left(s + \frac{\log_2 \Gamma}{\Gamma} s\right) \geq \left(1 + \frac{\log_2 \Gamma}{\Gamma}\right)^{\Gamma} f(s) \quad .$$

Further, by $1 + x \geq 2^x$ for $x \in [0, 1]$, the above is at least $2^{\log_2 \Gamma} f(s) = \Gamma f(s)$. So by the definition of t^* and the monotonicity of f, we get that

$$\frac{1}{2t^*} \leq \frac{\log_2 \Gamma}{\Gamma} s \quad .$$

The lemma then follows.

Consider the first time t after time 0 that the total size of the jobs waiting in the online algorithm drops to $\frac{1}{2}$. If $t \geq t^*$, then the fractional flow time is at least $\frac{\rho t^*}{2} \geq \frac{1}{4} \frac{\Gamma}{\log_2 \Gamma} \frac{\rho}{s}$. If $t < t^*$, then from time 0 to t^* the algorithm has processed jobs of total size at least $t^* s + \frac{1}{2}$. So the average speed in this period is $s + \frac{1}{2t^*}$ and the energy cost is at least

$$\begin{aligned} t^* f\left(s + \frac{1}{2t^*}\right) &= t^* \Gamma f(s) && \text{(by Eqn. (5.8))} \\ &= t^* \frac{\rho}{2} && \text{(by definition of } \rho) \\ &\geq \frac{1}{4} \frac{\Gamma}{\log_2 \Gamma} \frac{\rho}{s} && \text{(by Lemma 5.1).} \end{aligned}$$

In sum, the expected total cost of the algorithm is at least $\frac{1}{4} \frac{\Gamma}{\log_2 \Gamma} \frac{\rho}{s}$. So the competitive ratio is at least $\frac{1}{4} \frac{\Gamma}{\log_2 \Gamma}$.

We remark that the above lower bound instance only consists of two machines and two types of jobs with unit density, while our algorithm achieves asymptotically optimal competitive ratios for the most general settings of weighted fractional/integral flow-times with unrelated machines

References

[1] Susanne Albers, *Energy-efficient algorithms*, Communications of the ACM **53** (2010), no. 5, 86–96.

[2] Susanne Albers and Hiroshi Fujiwara, *Energy-efficient algorithms for flow time minimization*, ACM Transactions on Algorithms **3** (2007), no. 4.

[3] S. Anand, Naveen. Garg, and Amit Kumar, *Resource augmentation for weighted flow-time explained by dual fitting*, SODA, SIAM, 2012, pp. 1228–1241.

[4] Lachlan L.H. Andrew, Adam Wierman, and Ao Tang, *Optimal speed scaling under arbitrary power functions*, SIGMETRICS Performance Evaluation Review **37** (2009), no. 2, 39–41.

[5] Nikhil Bansal, Ho-Leung Chan, Tak-Wah Lam, and Lap-Kei Lee, *Scheduling for speed bounded processors*, ICALP (1), 2008, pp. 409–420.

[6] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs, *Speed scaling with an arbitrary power function*, SODA, SIAM, 2009, pp. 693–701.

[7] Nikhil Bansal, Kirk Pruhs, and Clifford Stein, *Speed scaling for weighted flow time*, SIAM Journal on Computing **39** (2009), no. 4, 1294–1308.

[8] Neal Barcelo, Daniel Cole, Dimitrios Letsios, Michael Nugent, and Kirk Pruhs, *Optimal energy trade-off schedules*, Sustainable Computing: Informatics and Systems (2013).

[9] Luiz André Barroso, *The price of performance*, ACM Queue **3** (2005), no. 7, 48–53.

[10] Niv Buchbinder and Joseph Naor, *The design of competitive online algorithms via a primal-dual approach*,

Foundations and Trends in Theoretical Computer Science **3** (2009), no. 2-3, 93–263.

[11] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V.N. Muralidhara, *A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation*, STOC, ACM, 2009, pp. 679–684.

[12] Nikhil R. Devanur, *Fisher markets and convex programs*, (2010), Manuscript, http://research.microsoft.com/en-us/um/people/nikdev/pubs/convex.pdf.

[13] Nikhil R. Devanur and Kamal Jain, *Online matching with concave returns*, STOC, 2012, pp. 137–144.

[14] Anapum Gupta, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, and Kirk Pruhs, *Scheduling heterogeneous processors isn't as easy as you think*, SODA, SIAM, 2012, pp. 1242–1253.

[15] Anapum Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs, *Scalably scheduling power-heterogeneous processors*, ICALP, Springer, 2010, pp. 312–323.

[16] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs, *Online primal-dual for non-linear optimization with applications to speed scaling*, Approximation and Online Algorithms, Springer, 2013, pp. 173–186.

[17] Tak-Wah Lam, Lap-Kei Lee, Isaac Kar-Keung To, and Prudence W.H. Wong, *Speed scaling functions for flow time scheduling based on active job count*, ESA, 2008, pp. 647–659.

[18] Kim Thang Nguyen, *Lagrangian duality in online scheduling with resource augmentation and speed scaling*, Proceedings of the 21st European Symposium on Algorithms, Springer, 2013, pp. 755–766.

[19] Andrew C.C. Yao, *Probabilistic computations: Toward a unified measure of complexity*, FOCS, IEEE, 1977, pp. 222–227.

A Convex conjugates and Fenchel duality

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is a function. The conjugate of f is a function $f^* : \mathbb{R} \to \mathbb{R}$ such that

$$f^*(\mu) := \sup_x \{\mu x - f(x)\}.$$

Although the conjugate is defined for any function f, for the rest of the article, we will assume that f is *strictly convex and differentiable*, since this is the case that is most interesting to the applications we discuss.

Properties of f^*:

- f^* is strictly convex and differentiable. (This property holds even if f is not strictly convex and differentiable.)
- $f^{**} = f$. (Here we use the assumption that f is strictly convex and differentiable.)
- If $g(x) = cf(x)$ for some constant c, then $g^*(\mu) = cf^*(\mu/c)$.
- If $g(x) = f(cx)$ for some constant c, then $g^*(\mu) = f^*(\mu/c)$.
- If $g(x) = f(x + a)$ for some constant a, then $g^*(\mu) = f^*(\mu) - \mu a$.
- If μ and x are such that $f(x) + f^*(\mu) = \mu x$ then $f'(x) = \mu$ and $(f^*)'(\mu) = x$.
- Vice versa, if $f'(x) = \mu$ then $(f^*)'(\mu) = x$ and $f(x) + f^*(\mu) = \mu x$.

We say that (x, μ) form a complementary pair wrt f if they satisfy one of the last two conditions stated above. We now calculate the conjugates of some simple strictly convex and differentiable functions for illustration.

- If $f(x) = \frac{1}{2}x^2$, then $f'(x) = x$. Thus $f^*(\mu)$ is obtained by letting $\mu = x$ in $\mu x - f(x)$, which is then equal to $\frac{1}{2}\mu^2$.
- If $f(x) = -\log(x)$, then $f'(x) = -1/x$. Set $\mu = -1/x$ to get $f^*(\mu) = -1 + \log(x) = -1 - \log(-\mu)$.
- Suppose $f(x) = x \log x$. Then $f'(x) = \log x + 1 = \mu$. So $x = e^{\mu-1}$. $f^*(\mu) = \mu x - f(x) = x(\log x + 1) - x \log x = x = e^{\mu-1}$. That is, $f^*(\mu) = e^{\mu-1}$.

Convex programs with linear constraints: Consider the following (primal) optimization problem.

$$\text{maximize} \sum_i c_i x_i - f_i(x_i) \quad \text{s.t.}$$
$$\forall j \ : \ \sum_i a_{ij} x_i \le b_j$$

We will derive a minimization problem that is the *dual* of this, using Lagrangian duality. This is usually a long calculation. The goal of this exercise is to identify a shortcut for the same.

Define the Lagrangian function

$$L(x, \lambda) := \sum_i c_i x_i - f_i(x_i) + \sum_j \lambda_j \left(b_j - \sum_i a_{ij} x_i \right).$$

We say that x is feasible if it satisfies all the constraints of the primal problem. Note that for all $\lambda \ge 0$ and x feasible, $L(x, \lambda) \ge \sum_i c_i x_i - f_i(x_i)$. Define the dual function

$$g(\lambda) = \max_x L(x, \lambda).$$

So for all λ, x, $g(\lambda) \ge L(x, \lambda)$. Thus $\min_{\lambda \ge 0} g(\lambda) \ge$ the optimum value for the primal program. The dual program is essentially $\min_{\lambda \ge 0} g(\lambda)$. We further simplify it as follows. Rewriting the expression for L,

$$L = \sum_i \mu_i x_i - f_i(x_i) + \sum_j b_j \lambda_j$$

where $\mu_i = c_i - \sum_j a_{ij}\lambda_j$. Now note that

$$\begin{aligned} g(\lambda) &= \max_x L(x,\lambda) \\ &= \max_x \left\{ \sum_i \mu_i x_i - f_i(x_i) \right\} + \sum_j b_j \lambda_j \\ &= \sum_i f_i^*(\mu_i) + \sum_j b_j \lambda_j \quad . \end{aligned}$$

Thus we get the dual optimization problem:

$$\begin{aligned} \text{minimize } & \sum_j b_j \lambda_j + \sum_i f_i^*(\mu_i) \quad \text{s.t.} \\ \forall i \ : \quad & \sum_j a_{ij}\lambda_j = c_i - \mu_i \\ \forall j \ : \quad & \lambda_j \geq 0 \end{aligned}$$

Note the similarity to LP duality. The differences are as follows. Suppose the concave part of the primal objective is $-\sum_i f_i(x_i)$. There is an extra variable μ_i for every variable x_i that occurs in f. In the constraint corresponding to x_i, $-\mu_i$ appears on the RHS along with the constant term. Finally the dual objective has $\sum_i f_i^*(\mu_i)$ in addition to the linear terms. In other words, we *relax* the constraint corresponding to x_i by allowing a slack of μ_i, and *charge* $\sum_i f_i^*(\mu_i)$ to the objective function.

Similarly, suppose we start with the primal problem

$$\begin{aligned} \text{maximize } & \sum_i c_i x_i - f_i(x_i) \quad \text{s.t.} \\ \forall j \ : \quad & \sum_i a_{ij} x_i \leq b_j \\ \forall i \ : \quad & x_i \geq 0 \end{aligned}$$

Then the dual problem is

$$\begin{aligned} \text{minimize } & \sum_j b_j \lambda_j + \sum_i f_i^*(\mu_i) \quad \text{s.t.} \\ \forall i \ : \quad & \sum_j a_{ij}\lambda_j \geq c_i - \mu_i \\ \forall j \ : \quad & \lambda_j \geq 0 \end{aligned}$$

As we saw, the optimum for the primal program is lower than the optimum for the dual program (weak duality). In fact, if the primal constraints are strictly feasible, that is there exist x_i such that for all j $\sum_i a_{ij} x_{ij} < b_j$, then the two optima are the same (strong duality) and the following generalized complementary slackness conditions characterize them:

- $x_i > 0 \Rightarrow \sum_j a_{ij}\lambda_j = c_i - \mu_i$;
- $\lambda_j > 0 \Rightarrow \sum_i a_{ij} x_i = b_i$; and
- x_i and μ_i form a complementary pair w.r.t. f_i, i.e., $\mu_i = f_i'(x_i)$, $x_i = (f^*)_i'(\mu_i)$ and $f_i(x_i) + f_i^*(\mu_i) = \mu_i x_i$.

Similarly suppose we start from a minimization problem of the form

$$\begin{aligned} \text{minimize } & \sum_i c_i x_i + f_i(x_i) \quad \text{s.t.} \\ \forall j \ : \quad & \sum_i a_{ij} x_i \geq b_j \\ \forall i \ : \quad & x_i \geq 0 \end{aligned}$$

Then the dual of this is

$$\begin{aligned} \text{maximize } & \sum_j b_j \lambda_j - \sum_i f_i^*(\mu_i) \quad \text{s.t.} \\ \forall i \ : \quad & \sum_j a_{ij}\lambda_j \leq c_i + \mu_i \\ \forall j \ : \quad & \lambda_j \geq 0 \end{aligned}$$