Resource Augmentation for Weighted Flow-time explained by Dual Fitting

S. Anand [*] Naveen Garg [*] Amit Kumar [*]

Abstract

We propose a general dual-fitting technique for analyzing on-line scheduling algorithms in the unrelated machines setting where the objective function involves weighted flow-time, and we allow the machines of the on-line algorithm to have $(1+\varepsilon)$-extra speed than the offline optimum (the so-called speed augmentation model). Typically, such algorithms are analyzed using non-trivial potential functions which yield little insight into the proof technique. We propose that one can often analyze such algorithms by looking at the dual (or Lagrangian dual) of the linear (or convex) program for the corresponding scheduling problem, and finding a feasible dual solution as the on-line algorithm proceeds. As representative cases, we get the following results :

- For the problem of minimizing weighted flow-time, we show that the greedy algorithm of Chadha-Garg-Kumar-Muralidhara is $O\left(\frac{1}{\varepsilon}\right)$-competitive. This is an improvement by a factor of $\frac{1}{\varepsilon}$ on the competitive ratio of this algorithm as analyzed by them.
- For the problem of minimizing weighted ℓ_k norm of flow-time, we show that a greedy algorithm gives an $O\left(\frac{k}{\varepsilon^{2+\frac{1}{k}}}\right)$-competitive ratio. This marginally improves the result of Im and Moseley.
- For the problem of minimizing weighted flow-time plus energy, and when the energy function $f(s)$ is equal to $s^\gamma, \gamma > 1$, we show that a natural greedy algorithm is $O(\gamma^2)$-competitive. Prior to our work, such a result was known for the related machines setting only (Gupta-Krishnaswamy-Pruhs).

1 Introduction

The problem of online scheduling of jobs on multiple heterogeneous machines has been well-studied in scheduling communities. A natural measure of performance is to consider the average weighted flow-time of jobs. The flow-time of a job is the total amount of time it spends in the system, i.e., the difference between its completion time and release date. For practical reasons, it is not desirable to migrate jobs across machines. However, pre-emption of jobs is a much needed assumption. Indeed there are strong lower bounds on the competitive ratio of any on-line algorithm if we do not allow pre-emption [14].

In the setting of unrelated machines, where the time required to process a job j on machine i equals p_{ij} (and this could be an arbitrary quantity), Garg and Kumar [10] showed that no on-line algorithm with bounded competitive ratio is possible. In fact, their example had only 3 machines, and jobs that had unit processing time on 2 of the 3 machines.

In light of such strong lower bounds, Chadha et al. [6] considered this problem in the speed augmentation model. In this model, we allow each machine in the on-line algorithm ε-fraction more speed than the corresponding machine in the offline algorithm. In other words, the work that a machine in the offline setting can do in 1 unit of time will require only $\frac{1}{1+\varepsilon}$ time on the corresponding machine in the on-line setting. They gave an $O(\frac{1}{\varepsilon^2})$-competitive algorithm for this problem. Their algorithm was greedy in nature : when a job arrives, it is dispatched immediately to the machine for which the *increase* in the weighted flow-time is smallest.

The objective function of minimizing the weighted sum of flow-time has the disadvantage that it can be very unfair to some jobs. One way of improving the fairness criteria is to look at weighted ℓ_k norm of flow-time of the jobs for some $k \geq 1$. Im and Moseley [12] gave an $O\left(\frac{k}{\varepsilon^{2+\frac{2}{k}}}\right)$-competitive algorithm for the problem minimizing weighted ℓ_k norm of flow-time in the unrelated machines model. However, their algorithm was based on a non-intuitive potential function.

Another recent line of research in this direction has been to allow the machines to run at variable speeds. A machine running at speed s incurs an energy cost of $f(s)$, where f is assumed to be a well-behaved function. Initial work in this area assumed f to be of the form $s^\gamma, \gamma > 1$, and more recent work has allowed f to be any increasing function (with some mild constraints). This framework models modern processors, and also the fact that multi-core architectures have the flexibility of changing the number of processors which are running at any time (see e.g., [11] and the references therein). A natural objective in this setting is to minimize the sum of weighted flow-time and energy consumed. In this setting, when $f(s) = s^\gamma$, Bansal et al. [3] gave an $O(\frac{\gamma}{\log \gamma})$-competitive algorithm for this problem. Gupta et al. [11] extended this to the case of related machines

[*]Department of Computer Science and Engineering, IIT Delhi, email : {anand,naveen,amitk}@cse.iitd.ac.in

and gave an $O(\gamma^2)$-competitive algorithm. Their results hold for more general class of energy functions, and they need the notion of speed augmentation. By giving extra speed to the processors, one can hope to come within a constant factor of the optimal flow-time, and increasing the speed slightly leads to only a modest increase in energy consumption.

Analyzing these algorithms turns out to be quite tricky. When one does not allow speed augmentation, one needs a very strong guarantee from any on-line algorithm: at every point of time t, the number (or total weight) of jobs waiting at any time in the on-line algorithm must be close to that in the off-line optimum. Indeed, if at time t, the gap becomes large, then the adversary can make the on-line algorithm pay a lot by maintaining this gap subsequently. However, with speed augmentation, any temporary build-up in the queue (as compared to off-line optimum) can be depleted with time. The most powerful approach for analyzing such algorithms has been to design clever potential functions, and show that these algorithms behave well in an amortized sense. Designing such potential functions is quite non-trivial, and often, yields little insight about how to design such functions for related problems. In this paper, we give a more direct approach for analyzing such algorithms.

All of these problems have a linear programming (or convex programming) relaxation. In this paper, we argue that often, it is much easier (and intuitive) to analyze these algorithms by dual fitting, i.e., show that there is a feasible solution to the corresponding dual LP (or Lagrangian relaxation) whose objective value is close to that of the on-line algorithm. In fact, we strongly believe that this method is as powerful as the potential function approach, and gives a much more principled approach towards analyzing such algorithms. As examples of our claim, we prove the following results :

- For the problem of minimizing weighted flow-time in the unrelated setting, we show that the greedy algorithm of Chadha et. al [6] is $O\left(\frac{1}{\varepsilon}\right)$-competitive. This is an improvement by a factor of $\frac{1}{\varepsilon}$ on the competitive ratio of this algorithm as analyzed by [6].

- For the problem of minimizing weighted ℓ_k norm of flow-time, we show that a greedy algorithm along the lines of [6] gives an $O\left(\frac{k}{\varepsilon^{2+\frac{1}{k}}}\right)$-competitive ratio. This marginally improves the result of Im and Moseley [12].

- For the problem of minimizing weighted flow-time plus energy in the unrelated setting and when the energy function $f(s)$ is equal to $s^\gamma, \gamma \geq 1$, we show that a natural greedy algorithm is $O(\gamma^2)$-competitive. Prior to our work, such a result was known for the related machines setting only [11]. Our analysis also works for a more general class of energy functions, but this is not the emphasis of our result.

We emphasize that although our techniques either improve or match the best known results for these problems, the main contribution of this paper is the technique of analyzing such algorithms by dual fitting. To illustrate the ideas involved in some more detail, consider the first problem above : minimizing weighted flow-time in the unrelated setting. The LP relaxation for this problem is given in Figure 1. The constraint (4.3) requires that the job j should be processed completely. Corresponding to this constraint we have a dual variable α_j. One interpretation of dual variables is the following : suppose we violate the corresponding constraint in the primal LP by a tiny amount, say δ. Then the change in the objective function is roughly $\alpha_j \delta$. Motivated by this, we think of the change in the objective function if we change the right hand side of constraint (4.3) from 1 to 0. Assuming no job is released after j, we should define α_j as the increase in the flow-time of all the jobs because of the arrival of j (see Section 4.2 for more details). Similarly, the constraint (4.4) says that in any unit time slot, a machine can process unit volume. If we change the right hand side from 1 to 0, and assume that no job arrives after time t, then the change in the objective function is exactly equal to the total weight of jobs waiting at time t – so, we define the corresponding dual variable β_{it} as equal to this quantity (scaled by some factor). The only remaining thing is to check that these dual variables are feasible. In the other settings considered in this paper, one can similarly define dual variables in a very natural manner. The bulk of the work lies in checking feasibility of the dual variables defined in this manner – but the approach remains very intuitive and principled.

In Section 4, we consider the problem of minimizing total weighted flow-time. We extend the result to the problem of minimizing weighted ℓ_k norm of the flow-time in Section 5. The problem of minimizing the sum of weighted flow-time and energy consumed is discussed in Section 6.

2 Related Work

The on-line problem of minimizing weighted flow-time has no bounded competitive ratio. In fact, this lower bound result holds even for the parallel machines setting [7]. Better results are known for the setting of unweighted flow-time (see e.g., [9] and the references therein). However, Garg and Kumar [10] showed that strong lower bounds on the competitive ratio hold even if all machines are identical and a job can go on a subset of machines only.

Speed augmentation in the online setting was first considered by Kalyanasundaram and Pruhs [13] who used it to get $O(\varepsilon^{-1})$-competitive algorithm for minimizing flow-time on a single machine in the non-clairvoyant setting. In the clairvoyant setting, Bansal and Pruhs [2] showed that SRPT (Shortest Remaining Processing Time first) is $O(\varepsilon^{-1})$ competitive for all ℓ_k-norms of flow time, $k \geq 1$. They extended this to weighted ℓ_k norms and showed that HDF (Highest Density First) is $O(\varepsilon^{-1})$-competitive [4]. Chekuri et al. [8] showed that the immediate dispatch algorithm of Avrahamai and Azar [1] is $O(\varepsilon^{-1})$-competitive for all ℓ_k-norms, $k \geq 1$. Garg and Kumar [6] gave a natural greedy algorithm for the setting of unrelated machines. They showed that this algorithm is $O(\varepsilon^{-2})$-competitive for weighted flow-time. Im and Moseley [12] extended this to an $O(k\varepsilon^{-(2+\frac{2}{k})})$-competitive algorithm for the case of weighted ℓ_k norm for all $k \geq 1$. Their algorithm, however, is based on a non-trivial potential function.

For the objective of weighted flow-time plus energy on a single machine, Bansal et al. [5] gave $O\left(\frac{\gamma}{\log \gamma}\right)$-competitive algorithm for fractional weighted flow plus energy, which then implied $O\left(\frac{\gamma^2}{\log^2 \gamma}\right)$-competitive algorithm for integral weighted flow plus energy. Here, the energy function was of the form s^γ. The scheduling algorithm was as follows : at any point of time, the speed s is chosen such that the energy consumed is equal to the total fractional weight of jobs waiting in the queue. Further, the machine followed the HDF scheduling policy. Recently, Bansal et.al. [3] considered a very general class of energy functions, and gave a constant factor competitive algorithm for the sum of fractional weighted flow-time and energy. This implies an $O\left(\frac{\gamma}{\log \gamma}\right)$-competitive algorithm for the sum of integral weighted flow-time and energy when the energy function is s^γ. Gupta et al. [11] extended this to the setting of related machines (the competitive ratio was $O(\gamma^2)$).

3 Preliminaries

We are given a set of m machines, and a set of jobs arrive over time. A job j is released at time r_j and requires p_{ij} units of processing if scheduled on machine i. Further, job j has a weight w_{ij} if it is scheduled on machine i. Let $d_{ij} = w_{ij}/p_{ij}$ denote the *density* of job j on machine i. A machine can process jobs preemptively, but we do not allow jobs to be migrated across machines. The flow-time of a job is the difference between its completion time and release date. If a job j gets processed on machine i, its weighted flow-time is w_{ij} times its flow-time.

4 Weighted flow-time on Unrelated Machines

We consider the problem of minimizing the weighted sum of flow-time of jobs. In Section 4.1, we describe the greedy algorithm of Chadha et al. [6]. In Section 4.2, we analyze this algorithm. We first describe the LP relaxation and its dual. Then we show that the dual variables can be set as the jobs arrive. These dual variables need to satisfy two conditions : (i) they should be feasible to the dual LP, and (ii) the dual objective value should be close to the weighted flow-time of the greedy algorithm.

4.1 The greedy algorithm

When a job arrives, the algorithm dispatches it immediately to one of the machines. Let $\mathcal{A}_i(t)$ the set of jobs which get dispatched to machine i and are unfinished at time t. Consider a job $j \in \mathcal{A}_i(t)$. Let $p_j(t)$ denote the remaining processing time of job j. Define the *residual density* of j at time t, $d_j(t)$, as $\frac{w_{ij}}{p_j(t)}$. Each machine follows the highest residual density first (HRDF) algorithm : at any time t, machine i processes the job in $\mathcal{A}_i(t)$ with highest residual density. Observe that if all weights are 1, then the HRDF policy becomes SRPT (Shortest Remaining Processing Time first) policy. Also, it is worth noting, that arrival of new jobs does not change the residual density of a job, and hence, the relative ordering of jobs does not change with time.

It remains to specify the dispatch policy. When a job j arrives at time t, we compute, for each machine i, the increase in weighted flow-time if we dispatch j to machine i. More precisely, let Q_{ij} denote the quantity

$$\frac{1}{1+\varepsilon}\left\{ w_{ij} \cdot \sum_{j' \in \mathcal{A}_i(t): d_{j'}(t) \geq d_{ij}} p_{j'}(t) + w_{ij} p_{ij} \right.$$

$$\left. + \, p_{ij} \cdot \sum_{j' \in \mathcal{A}_i(t): d_{j'}(t) < d_{ij}} w_{ij'} \right\}. \tag{4.1}$$

The dispatch algorithm is greedy : it assigns j to the machine i for which Q_{ij} is minimum. We shall use $\mathcal{A}$ to denote this algorithm. We shall denote the weighted flow-time of $\mathcal{A}$ by $F^{\mathcal{A}}$.

4.2 Analysis

We first write an LP relaxation for the problem of minimizing weighted flow-time. We shall use the letters

Primal LP	Dual LP
(4.2) $\min \sum_{i,j,t} w_{ij} \cdot x_{ijt} \cdot \left(\frac{t - r_j}{p_{ij}} + \frac{1}{2} \right)$	(4.5) $\max \sum_j \alpha_j - \sum_{i,t} \beta_{it}$
(4.3) $\sum_{i,t} \frac{x_{ijt}}{p_{ij}} = 1 \quad$ for all j	(4.6) $\frac{\alpha_j}{p_{ij}} - \beta_{it} \le d_{ij}(t - r_j) + \frac{w_{ij}}{2} \quad$ for all i, j, t
(4.4) $\sum_j x_{ijt} \le 1 \quad$ for all i, t	(4.7) $\alpha_j, \beta_{it} \ge 0 \quad$ for all i, j, t
$x_{ijt} \ge 0 \quad$ for all j, i, $t \ge r_j$	

Figure 1: The LP relaxation and its dual

i, j, t for machines, jobs, and (integral) time respectively. We assume all release dates r_j and processing times p_{ij} are integers. We divide the time-line into *slots* of unit length, i.e., intervals of the form $[t, t+1]$ for all non-negative integers t. Let x_{ijt} denote the fraction of the slot $[t, t+1]$ used by machine i for processing job j. Note that x_{ijt} is defined only if $t \ge r_j$. The LP relaxation is described in Figure 1.

The quantity $\sum_t \frac{x_{ijt}}{p_{ij}}$ refers to the fractional amount of job j processed on machine i. Thus, constraint (4.3) refers to the fact that job j is processed to a fraction of one. Constraint (4.4) refers to the fact that a machine can only perform one unit of processing during $[t, t+1]$. In the objective function, the first term can be thought of as fractional flow-time – $\frac{x_{ijt}}{p_{ij}}$ fraction of the job j spends $(t - r_j)$ unit of time before it finishes. The second term (ignoring the factor of 1/2) in the objective function measures the total processing time – this term is required because a solution to the LP above could process a job simultaneously on multiple machines, and so, the flow-time of a job could be much smaller than its processing time. The following claim, which says that the LP above is indeed a relaxation, was proved in [9].

Claim 4.1. *[9] For a (non-migratory) schedule $\mathcal{S}$, consider the corresponding feasible solution to the LP above. Then the objective value of this solution is at most the weighted flow-time of $\mathcal{S}$.*

The dual LP of this relaxation is described in Figure 1. The goal is to prove the following theorem.

Theorem 4.1. *There exists a feasible solution α_j, β_{it} to the dual LP such that the objective value of this solution, i.e., $\sum_j \alpha_j - \sum_{i,t} \beta_{it}$, is $\Omega(\varepsilon \cdot F^{\mathcal{A}})$.*

We now define these dual variables. The variable β_{it} is equal to $\frac{1}{1+\varepsilon}$ times the total weight of jobs which are waiting at time t on machine i, i.e.,

$$\beta_{it} = \frac{1}{1+\varepsilon} \sum_{j \in \mathcal{A}_i(t)} w_{ij}.$$

Observe that $\sum_{i,t} \beta_{it}$ is exactly equal to $\frac{F^{\mathcal{A}}}{1+\varepsilon}$. Now, we define α_j. Suppose j is released at time t and it gets assigned to machine i. Then α_j is equal to the increase in weighted flow-time of jobs in $\mathcal{A}_i(t)$ (and job j). In other words, α_j is equal to Q_{ij} as defined in equation (4.1).

Lemma 4.1. *$\sum_j \alpha_j - \sum_{i,t} \beta_{it}$ is at least $\frac{\varepsilon}{1+\varepsilon} \cdot F^{\mathcal{A}}$.*

Proof. The statement follows because $\sum_j \alpha_j$ is equal to $F^{\mathcal{A}}$ and $\sum_{i,t} \beta_{it}$ is equal to $\frac{F^{\mathcal{A}}}{1+\varepsilon}$. ■

It now remains to prove that this is a feasible solution.

Lemma 4.2. *The values $\alpha_j/2, \beta_{it}/2$ form a feasible solution to the dual LP.*

Proof. Fix a job j and machine i. Let t denote the release date of j and t' be an arbitrary time after t. We need to show that

$$\frac{\alpha_j}{p_{ij}} - \beta_{it'} \le d_{ij}(t' - t) + \frac{w_{ij}}{2}. \tag{4.8}$$

We can assume that no new job arrives after j – indeed, this will not affect the value α_j, and will only decrease $\beta_{it'}$. Let $\mathcal{A}_i^1(t)$ denote the set of jobs j' in $\mathcal{A}_i(t)$ for which $d_{j'}(t) \ge d_{ij}$, and $\mathcal{A}_i^2(t)$ be the jobs j' in $\mathcal{A}_i(t)$ for which $d_{j'}(t) < d_{ij}$. We arrange the jobs in $\mathcal{A}_i(t)$ in ascending order of residual density; let the sequence of jobs be $j_1, \ldots, j_n$. Note that machine i will process the jobs in $\mathcal{A}_i(t)$ in this sequence. Let the set $\mathcal{A}_i^1(t)$ consist of jobs $\{j_1, \ldots, j_r\}$, and (hence) the set $\mathcal{A}_i^2(t)$ be $\{j_{r+1}, \ldots, j_n\}$.
Suppose at time t', machine i is processing job j_k. Two cases arise :

- $j_k \in \mathcal{A}_i^1(t)$: Here, $(t'-t)(1+\varepsilon) \geq \sum_{l=1}^{k'-1} p_{j_l}(t)$. Further, all jobs among $j_k, \ldots, j_n$ belong to $\mathcal{A}_i(t')$. Recall that $\alpha_j = \min_{i'} Q_{i'j} \leq Q_{ij}$. So,

$$\begin{aligned}\frac{(1+\varepsilon)\alpha_j}{p_{ij}} &\leq \frac{(1+\varepsilon)Q_{ij}}{p_{ij}} \\ &\leq \sum_{j' \in A_i^1(t)} d_{ij} p_{j'}(t) + w_{ij} + \sum_{j' \in \mathcal{A}_i^2(t)} w_{ij'} \\ &= d_{ij}\left(\sum_{l=1}^{k-1} p_{j_l}(t)\right) + d_{ij}\left(\sum_{l=k}^{r} p_{j_l}(t)\right) + w_{ij} \\ &\quad + \sum_{j' \in \mathcal{A}_i^2(t)} w_{ij'} \\ &\leq d_{ij}(t'-t)(1+\varepsilon) + \sum_{l=k}^{r} w_{ij_l} + w_{ij} + \sum_{j' \in \mathcal{A}_i^2(t)} w_{ij'} \\ &\leq d_{ij}(t'-t)(1+\varepsilon) + w_{ij} + (1+\varepsilon)\beta_{it'}\end{aligned}$$

where the second last inequality follows from the fact that if $j' \in \mathcal{A}_i^1(t)$, then $d_{j'}(t) \geq d_{ij}$, and so $p_{j'}(t) d_{ij} \leq w_{ij'}$.

- $j_k \in \mathcal{A}_i^2(t)$: Again, $(t'-t)(1+\varepsilon) \geq \sum_{l=1}^{k'-1} p_{j_l}(t)$, and all jobs among $j_k, \ldots, j_n$ belong to $\mathcal{A}_i(t')$. As above,

$$\begin{aligned}\frac{(1+\varepsilon)\alpha_j}{p_{ij}} &\leq \frac{(1+\varepsilon)Q_{ij}}{p_{ij}} \\ &\leq d_{ij}\left(\sum_{j' \in A_i^1(t)} p_{j'}(t)\right) + w_{ij} + \sum_{j' \in \mathcal{A}_i^2(t)} w_{ij'} \\ &= d_{ij}\left(\sum_{l=1}^{r} p_{j_l}(t)\right) + w_{ij} + \sum_{l=r+1}^{k-1} w_{ij_l} + \sum_{l=k}^{n} w_{ij_l} \\ &\leq d_{ij}\left(\sum_{l=1}^{r} p_{j_l}(t)\right) + w_{ij} + d_{ij}\left(\sum_{l=r+1}^{k-1} p_{j_l}(t)\right) \\ &\quad + (1+\varepsilon)\beta_{it'} \\ &\leq d_{ij}(t'-t)(1+\varepsilon) + w_{ij} + (1+\varepsilon)\beta_{it'}\end{aligned}$$

Thus, we have shown that $\frac{\alpha_j}{2}, \frac{\beta_{it}}{2}$ satisfy the inequality (4.8) the in both cases. ∎

Proof of Theorem 4.1. The desired dual feasible solution is $\alpha_j/2, \beta_{it}/2$ (Lemma 4.2). Lemma 4.1 shows that the dual objective value is at least $\frac{\varepsilon}{2(1+\varepsilon)} F^{\mathcal{A}} = \Omega(\varepsilon) \cdot F^{\mathcal{A}}$. ∎

5 Extension to ℓ_k norm

We now extend our result to weighted ℓ_k norm of the flow-times of jobs. We state the objective function more precisely. For a schedule $\mathcal{S}$, let C_j be the completion time of job j. Define the weighted ℓ_k-norm of the flow-time vector as

$$\left(\sum_j w_{i(j)j}(C_j - r_j)^k\right)^{\frac{1}{k}},$$

where $i(j)$ is the machine on which j gets scheduled. It will be easier to deal with the k^{th} power of this quantity. Let $F^{\mathcal{S},k}$ denote $\sum_j w_{i(j)j}(C_j - r_j)^k$.

5.1 Scheduling Algorithm

We define the scheduling algorithm in two stages – the first algorithm $\mathcal{A}$ is given a speed-up of $(1+\varepsilon)$, and then the schedule $\mathcal{B}$ is given a speedup of $(1+\varepsilon)$ over $\mathcal{A}$.

Our algorithm $\mathcal{A}$ again behaves in a manner similar to the greedy algorithm described in Section 4.1. When a job arrives, it gets immediately dispatched to one of the machines, and hence, we can define the quantities $\mathcal{A}_i(t), p_j(t), d_j(t)$ as before. Again, each machine follows the highest residual density first (HRDF) algorithm. It remains to describe the dispatch policy.

For a job $j' \in \mathcal{A}_i(t)$, let $R_{j'}(t)$ be the remaining time before j' finishes. i.e., $C_{j'} - t$, where $C_{j'}$ is the completion time of j' assuming no jobs arrive after time t. Suppose a job j arrives at time t. Given a machine i, define Q_{ij} as

$$(5.9) \qquad w_{ij} \cdot \left(\frac{\sum_{j' \in \mathcal{A}_i(t): d_{j'}(t) \geq d_{ij}} p_{j'}(t) + p_{ij}}{1+\varepsilon}\right)^k + \sum_{j' \in \mathcal{A}_i(t): d_{j'}(t) < d_{ij}} w_{ij'} \cdot \left(\left(R_{j'}(t) + \frac{p_{ij}}{1+\varepsilon}\right)^k - R_{j'}(t)^k\right).$$

Now, assign j to the machine i for which Q_{ij} is smallest. Note that this is different from the increase in $F^{\mathcal{A},k}$ if we assign j to machine i – it does not look at the *age* of a job. The quantity is just measuring the increase in flow-time if we assume all jobs in $\mathcal{A}_i(t)$ are released at time t. Clearly, this cannot yield a competitive algorithm because if the age of a job is very high, then we should process it soon. However, we will show that after giving one more round of extra speed to the machines, the resulting schedule becomes constant-competitive.

Before we describe the schedule $\mathcal{B}$, we note one property of the HRDF policy. Fix a machine i, and consider the jobs which get dispatched to i in $\mathcal{A}$ – the HRDF policy gives a total ordering on these jobs. Indeed, consider two such jobs j_1 and j_2 which are dispatched to i. Let t denote $\max(r_{j_1}, r_{j_2})$. Then HRDF prefers j_1 if $d_{j_1}(t) \geq d_{j_2}(t)$, otherwise it prefers j_2. It is easy to show that this gives a total ordering on the jobs dispatched to i, and the HRDF policy processes the jobs according to this ordering.

In the schedule $\mathcal{B}$, the set of jobs dispatched to any machine i will be the same as in $\mathcal{A}$. Fix a machine i. In $\mathcal{B}$, the jobs will be prioritized by the ordering given by HRDF policy in $\mathcal{A}$ (as described above). Further the machine i will process at $(1+\varepsilon)$-times the speed of i in $\mathcal{A}$.

5.2 Analysis

We first establish some useful inequalities.

Lemma 5.1. *Suppose $a_1, \ldots, a_m$ are non-negative integers. Let b_i denote $a_1 + \cdots + a_i$. Then,*

$$(5.10)\quad k \cdot \sum_{i=1}^{m-1} a_{i+1} b_i^{k-1} \le b_m^k \le k \cdot \sum_{i=1}^{m} a_i b_i^{k-1}.$$

Taking $a_1 = \cdots = a_m = 1$, we get

$$(5.11)\qquad k \cdot \sum_{i=1}^{m-1} i^{k-1} \le m^k \le k \cdot \sum_{i=1}^{m} i^k.$$

Proof. Consider the area under the curve $y = x^{k-1}$ where x varies from 0 to b_m. Then, it is easy to check that this area is equal to $\frac{1}{k} b_m^k$. Now, one way of upper bounding this area is the following : for each $i = 1, \ldots, m$, construct a rectangle of height b_i^{k-1} and base between the points b_{i-1} and b_i on the x-axis. It is easy to check that the sum of the areas of these rectangles is at least the area under the curve mentioned above. The lower bound is proved similarly. ■

Lemma 5.2. *Let a, b be non-negative integers. Then, for any ε, $0 < \varepsilon < 1$,*

$$(5.12)\quad (a+b)^k \le (1+\varepsilon)a^k + \left(\frac{3k}{\varepsilon}\right)^k \cdot b^k.$$

Proof. If $b \le \frac{\varepsilon \cdot a}{2k}$, then $(a+b)^k \le (1+\frac{\varepsilon}{2k})^k a^k \le (1+\varepsilon)a^k$. Otherwise, $(a+b)^k \le \left(\frac{2k}{\varepsilon}+1\right)^k b^k$. ■

We now write an LP relaxation for this problem. Here, it will be convenient to think of time t as a continuous variable. The LP relaxation is described in Figure 2. The variable x_{ijt} should now be thought of as the rate at which machine i is processing job j at time t. The constraints are same as those in the LP for total weighted flow-time case (Figure 1).
We now argue that it is indeed a relaxation.

Lemma 5.3. *Consider a schedule $\mathcal{S}$ (where machines run at their original speed), and let x_{ijt} be the corresponding LP solution. Then, the objective value of this LP solution is at most $2F^{\mathcal{S},k}$.*

Proof. Consider a job j, and suppose its completion time is C_j. Then its contribution to $F^{\mathcal{S},k}$ is $w_{ij}(C_j - r_j)^k$ – here i is the machine on which j gets scheduled. Now,

$$\int_t \frac{w_{ij} x_{ijt}}{p_{ij}} (t-r_j)^k dt \le w_{ij}(C_j - r_j)^k \int_t \frac{x_{ijt}}{p_{ij}} dt = w_{ij}(C_j - r_j)^k.$$

Thus the first term in the objective function is at most $F^{\mathcal{A},k}$. Now, if a job j is scheduled on machine i by $\mathcal{S}$, then $\int_t x_{ijt} dt = p_{ij}$, and so, $\int_t w_{ij} p_{ij}^{k-1} x_{ijt} dt = w_{ij} p_{ij}^k \le w_{ij}(C_j - r_j)^k$. Therefore, the second term in the objective function is also at most $F^{\mathcal{A},k}$. ■

Rest of the plan of the proof is to come up with a feasible dual solution whose objective value is close to $F^{\mathcal{A},k}$. The dual LP is shown in Figure 2. We assume that the machines in $\mathcal{A}$ have $(1+\varepsilon)$-extra speed. For a machine i and time t, define β_{it} as

$$k \cdot \sum_{j \in \mathcal{A}_i(t)} w_{ij} R_j(t)^{k-1}.$$

For a job j, define α_j as the quantity Q_{ij} defined in in (5.9) where i is the machine to which j gets dispatched. The following lemma shows that the dual variables as defined above have objective value close to the cost of schedule $\mathcal{B}$.

Lemma 5.4. *The objective value for the solution α_j, β_{it} defined above, i.e., $\sum_j \alpha_j - \sum_i \int_t \beta_{it} dt$ is $\Omega\left(\varepsilon^{k+1} F^{\mathcal{B},k}\right)$.*

Proof. Note that

$$(5.19)\qquad \sum_i \int_t \beta_{it} = k \sum_i \int_t \sum_{j \in \mathcal{A}_i(t)} w_{ij} R_j(t)^{k-1} dt = k \sum_j w_{i(j)j} \int_{t \ge r_j} R_j(t)^{k-1} dt,$$

where $i(j)$ is the machine to which j gets dispatched. For a fixed j, let us try to understand the right hand side above. Let $t_0 = r_j < t_1 < \ldots < t_r$ be the times at which $R_j(t)$ is discontinuous – this happens precisely when a job of higher density than j gets dispatched to machine $i(j)$ at time t. If t is not one of such times, then $R_j(t)$ decreases linearly at the rate of $1+\varepsilon$. So

Primal LP	Dual LP
(5.13) $\min \sum_{i,j} w_{ij} \int_t \left(\frac{x_{ijt}}{p_{ij}} (t - r_j)^k + p_{ij}^{k-1} x_{ijt} \right) dt$	(5.16) $\max \sum_j \alpha_j - \sum_i \int_t \beta_{it} dt$
(5.14) $\sum_i \int_{t \geq r_j} \frac{x_{ijt}}{p_{ij}} dt = 1 \quad \forall j$	(5.17) $\frac{\alpha_j}{p_{ij}} - \beta_{it} \leq d_{ij}(t - r_j)^k + w_{ij} p_{ij}^{k-1} \quad \forall i, j, t$
(5.15) $\sum_j x_{ijt} \leq 1 \quad \forall i, t$	(5.18) $\alpha_j, \beta_{it} \geq 0 \quad \forall i, j, t$
$x_{ijt} \geq 0 \quad \forall j, i, t \geq r_j$	

Figure 2: The LP relaxation and its dual

(define t_{r+1} as C_j),

$$\int_{t \geq r_j} k R_j(t)^{k-1} dt = \sum_{l=0}^{r} k \int_{t_l^+}^{t_{l+1}^-} R_j(t)^{k-1} dt$$

$$= k \sum_{l=0}^{r} \int_{t_l^+}^{t_{l+1}^-} \left(R_j(t_l^+) - (1+\varepsilon)(t - t_l) \right)^{k-1} dt$$

$$= \frac{1}{(1+\varepsilon)} \cdot \sum_{l=0}^{r} \left(R_j(t_l^+)^k - R_j(t_{l+1}^-)^k \right)$$

$$(5.20) \quad = \frac{1}{(1+\varepsilon)} \cdot \sum_{l=0}^{r} \left(R_j(t_l^+)^k - R_j(t_l^-)^k \right)$$

where we define $R_j(t_0^-)$ as 0. Let $\Delta(j)$ denote the quantity

$$\sum_{l=0}^{r} \left(R_j(t_l^+)^k - R_j(t_l^-)^k \right).$$

Now, observe that Q_{ij} as defined in (5.9) is exactly measuring the increase in $\sum_{j'} w_{i(j')j} \Delta(j)$ if j gets dispatched to machine i and no jobs arrive after j. Thus, $\sum_j \alpha_j$ is exactly equal to $\sum_j w_{i(j)j} \Delta_j$. Combining this with (5.19) and (5.20), we see that

$$\sum_j \alpha_j - \sum_i \int_t \beta_{it} = \frac{\varepsilon}{1+\varepsilon} \sum_j \Delta_j.$$

It remains to prove that $\Delta_j \geq \varepsilon^k F_j^{\mathcal{B},k}$, where $F_j^{\mathcal{B},k}$ denotes the k^{th} power of the flow-time of j in the schedule $\mathcal{B}$.

We begin with a useful claim first.

Claim 5.1. *For a job j and time $t \geq r_j$, if $R_j(t) \leq \varepsilon(t - r_j)$, then job j finishes by time t in $\mathcal{B}$.*

Proof. For the sake of this proof, let $R_j^{\mathcal{A}}(t)$ denote the remaining processing time of j at time t in $\mathcal{A}$, i.e., $R_j(t)$, and $R_j^{\mathcal{B}}(t)$ be the corresponding quantity for $\mathcal{B}$. It is easy to check the invariant that for any time t and job j, $R_j^{\mathcal{B}}(t) \leq R_j^{\mathcal{A}}(t)$. Now, at time r_j, let $V^{\mathcal{A}}$ be the total remaining processing time of jobs which have precedence over j (we have already mentioned that HRDF induces a total ordering on jobs). Define $V^{\mathcal{B}}$ similarly for the schedule $\mathcal{B}$. Our invariant implies that $V^{\mathcal{B}} \leq V^{\mathcal{A}}$. Suppose, for the sake of contradiction, that j does not finish at time t in $\mathcal{B}$. Then $\mathcal{B}$ processes jobs of priority higher than j during $[r_j, t]$, and the total processing done by $\mathcal{B}$ during this interval is $(1+\varepsilon)^2(t - r_j)$. If V denotes the total processing time of jobs of priority higher than j which get released during $[r_j, t]$, it follows that

$$V + V^{\mathcal{B}} > (1+\varepsilon)^2(t - r_j).$$

But then, the total remaining processing time of such jobs in $\mathcal{A}$ at time t will be at least

$$V + V^{\mathcal{A}} - (1+\varepsilon)(t - r_j) \geq V + V^{\mathcal{B}} - (1+\varepsilon)(t - r_j) > \varepsilon(t - r_j).$$

This implies that $R_j(t) > \varepsilon(t - r_j)$, a contradiction. ■

Fix a job j, and let $t_0 = r_j < t_1 < \ldots < t_r < t_{r+1} = C_j$ be as defined above (with respect to schedule $\mathcal{A}$). Let $F_j(t)$ be the flow-time of j assuming no jobs arrive after time t (in schedule $\mathcal{A}$). So, $F_j(t) = R_j(t) + (t - r_j)$. Let t be the first time in $\mathcal{A}$ such that $R_j(t) \leq \varepsilon(t - r_j)$. Since $R_j(t)$, as a function of t, is continuous except with finite discrete positive jumps, it must be the case that at time t, $R_j(t) = \varepsilon(t - r_j)$. Claim 5.1 shows that j will finish by time t in $\mathcal{B}$. Therefore,

$$\varepsilon^k F_j^{\mathcal{B},k} \leq \varepsilon^k (t - r_j)^k \leq R_j(t)^k$$

Suppose t lies between t_m and t_{m+1}. Then $R_j(t) \leq$

$R_j(t_m^+)$, and observe that

$$R_j(t_m^+)^k \le R_j(t_m^+)^k + \sum_{l=0}^{m-1} \left(R_j(t_l^+)^k - R_j(t_{l+1}^-)^k\right)$$
$$= \sum_{l=0}^{m} \left(R_j(t_l^+)^k - R_j(t_l^-)^k\right) \le \Delta_j.$$

■

Now, we prove that α_j, β_{it} are nearly feasible to the dual LP.

Lemma 5.5. *The values $\alpha_j/\gamma, \beta_{it}/\gamma$ form a feasible solution to the dual LP, where γ is $\left(\Omega\left(\frac{k}{\varepsilon}\right)\right)^k$.*

Proof. Fix a job j and machine i. Let t denote the release date of r_j, and fix a time $t' \ge t$. We will show that

$$\frac{\alpha_j}{p_{ij}} - \beta_{it'} \le \left(\Omega\left(\frac{k}{\varepsilon}\right)\right)^k \cdot d_{ij}((t'-t)+p_{ij})^k. \quad (5.21)$$

This will imply the lemma because

$$d_{ij}((t'-t)+p_{ij})^k \le 2^k d_{ij}(t'-t)^k + 2^k w_{ij} p_{ij}^{k-1}.$$

We will make one simplifying assumption – no new job arrivals happen during $[t,t']$ – indeed such job arrivals will not change α_j, but would only increase $\beta_{it'}$. Now, arrange the jobs j' in $\mathcal{A}_i(t)$ in descending order of $d_{j'}(t)$ values – let this order be $j_1, \dots, j_m$. We first observe that at time t', $d_{j_1}(t') \ge d_{j_2}(t') \ge \dots \ge d_{j_m}(t')$. Indeed, machine i processes jobs in this order. So at time t', if it is processing job j_l, then, $d_{j_1}(t') = \dots = d_{j_{l-1}}(t') = \infty$, $d_{j_l}(t') \ge d_{j_l}(t)$ and $d_{j_{l+1}}(t') = d_{j_{l+1}}(t), \dots, d_{j_m}(t') = d_{j_m}(t)$.

We develop some more notation. Let $\mathcal{A}_i^1(t)$ denote those jobs j' in $\mathcal{A}_i(t)$ for which $d_{j'}(t) \ge d_{ij}$, and $A_i^2(t)$ be the remaining set of jobs. Assume that $\mathcal{A}_i^1(t) = \{j_1, \dots, j_r\}$, and so, $\mathcal{A}_i^2(t) = \{j_{r+1}, \dots, j_m\}$.

Recall that $\alpha_j = \min_i Q_{ij}$. So,

$$\frac{\alpha_j}{p_{ij}} \le \frac{Q_{ij}}{p_{ij}} = d_{ij}\left(\frac{\sum_{j'\in\mathcal{A}_i^1(t)} p_{j'}(t) + p_{ij}}{1+\varepsilon}\right)^k$$
$$+ \frac{\sum_{j'\in\mathcal{A}_i^2(t)} w_{ij'}\left((R_{j'}(t) + \frac{p_{ij}}{1+\varepsilon})^k - R_{j'}(t)^k\right)}{p_{ij}} \quad (5.22)$$

Now, we simplify each of the terms above.

Claim 5.2.

$$d_{ij}\left(\frac{\sum_{j'\in\mathcal{A}_i^1(t)} p_{j'}(t) + p_{ij}}{1+\varepsilon}\right)^k$$
$$\le k \sum_{j'\in\mathcal{A}_i^1(t)} w_{ij'} R_{j'}(t')^{k-1}$$
$$+ \left(\frac{3k}{\varepsilon}\right)^k \cdot d_{ij}((t'-t)+p_{ij})^k.$$

Proof. During the period $[t,t']$, $(t'-t)(1+\varepsilon)$ amount of processing is done. So, $\sum_{j'\in\mathcal{A}_i^1(t)}(p_{j'}(t) - p_{j'}(t')) = \sum_{j'\in\mathcal{A}_i^1(t)} p_{j'}(t) - \sum_{j'\in\mathcal{A}_i^1(t')} p_{j'}(t') \le (1+\varepsilon)(t'-t)$. Also, observe that $R_{j_l}(t') = \frac{1}{1+\varepsilon}\sum_{s=1}^{l} p_{j_s}(t')$. Therefore,

$$\left(\frac{\sum_{j'\in\mathcal{A}_i^1(t)} p_{j'}(t) + p_{ij}}{1+\varepsilon}\right)^k$$
$$\le \left(\sum_{j'\in\mathcal{A}_i^1(t')} \frac{p_{j'}(t')}{1+\varepsilon} + p_{ij} + (t'-t)\right)^k$$
$$\overset{(5.12)}{\le} (1+\varepsilon)\left(\sum_{j'\in\mathcal{A}_i^1(t')} \frac{p_{j'}(t')}{1+\varepsilon}\right)^k$$
$$+ \left(\frac{3k}{\varepsilon}\right)^k (p_{ij} + (t'-t))^k$$
$$\overset{(5.10)}{\le} k \sum_{j'\in\mathcal{A}_i^1(t')} p_{j'}(t') R_{j'}(t')^{k-1} +$$
$$\left(\frac{3k}{\varepsilon}\right)^k (p_{ij} + (t'-t))^k.$$

Now, multiplying both sides by d_{ij}, and using the fact that $j' \in \mathcal{A}_i^1(t)$, we get $p_{j'}(t') d_{ij} \le p_{j'}(t) d_{ij} = w_{ij'}\frac{d_{ij}}{d_{j'}(t)} \le w_{ij'}$. This implies the lemma. ■

Bounding the second term in equation (5.22) is more non-trivial. We begin with the following claim.

Claim 5.3.

$$\sum_{j'\in\mathcal{A}_i^2(t)} \frac{w_{ij'}}{p_{ij}}\left(\left(R_{j'}(t) + \frac{p_{ij}}{1+\varepsilon}\right)^k - R_{j'}(t)^k\right)$$
$$\le \frac{k}{1+\frac{\varepsilon}{2}} \sum_{j'\in\mathcal{A}_i^2(t)} w_{ij'} R_{j'}(t)^{k-1} + 3\left(\frac{8k}{\varepsilon}\right)^k d_{ij} p_{ij}^k$$

Proof. Using binomial expansion,

$$\left(R_{j'}(t)+\frac{p_{ij}}{1+\varepsilon}\right)^k - R_{j'}(t)^k$$
$$= k\cdot\frac{p_{ij}}{1+\varepsilon}\cdot R_{j'}(t)^{k-1} + \sum_{l=2}^{k}\binom{k}{l}R_{j'}(t)^{k-l}\left(\frac{p_{ij}}{1+\varepsilon}\right)^l.$$

If $p_{ij} \le \frac{\varepsilon}{8k}R_{j'}(t)$, then it is easy to check that the above expression is at most $\frac{k}{1+\varepsilon/2}\cdot p_{ij}\cdot R_{j'}(t)^{k-1}$. So, assume that $R_{j'}(t) \le \frac{8k}{\varepsilon}p_{ij}$ – let J' denote the set of such jobs. For such jobs, it is again easy to check (using the binomial expansion) that

(5.23)
$$\sum_{j'\in J'}\frac{w_{ij'}}{p_{ij}}\left(\left(R_{j'}(t)+\frac{p_{ij}}{1+\varepsilon}\right)^k - R_{j'}(t)^k\right)$$
$$\le \frac{k}{1+\varepsilon}\sum_{j'\in J'} w_{ij'}R_{j'}(t)^{k-1} + 2\left(\frac{8k}{\varepsilon}\right)^{k-1}p_{ij}^{k-1}\sum_{j'\in J'}w_{ij'}$$

Observe that there is a job in J' (the one which is processed last by i) for which $(1+\varepsilon)R_{j'}(t) \ge \sum_{j'\in J'}p_{j'}(t)$. So, we get

$$\sum_{j'\in J'} w_{ij'} \le \sum_{j'\in J'} p_{j'}(t)d_{ij} \le \frac{8k(1+\varepsilon)}{\varepsilon}w_{ij},$$

where the first inequality follows from the fact that $j' \in \mathcal{A}_i^2(t)$. Substituting the above in (5.23) implies the lemma. ∎

We now simplify the expression in the claim above. Recall that the set of jobs in $\mathcal{A}_i(t)$ are ordered $j_1,\ldots,j_r$, and those in $\mathcal{A}_i^2(t)$ are $j_{r+1},\ldots,j_m$. Also, note that $(1+\varepsilon)R_{j_s}(t) = \sum_{s'=1}^{s}p_{j_{s'}}(t)$. Let ℓ be the highest index for which $R_{j_\ell}(t) \le \frac{16k(t'-t)}{\varepsilon}$. We now split the sum in the right hand side of Claim 5.3 in two parts, and analyze each of them.

Claim 5.4.

$$k\sum_{s=r+1}^{\ell} w_{ij_s}R_{j_s}(t)^{k-1} \le \left(\Omega\left(\frac{k}{\varepsilon}\right)\right)^k d_{ij}\cdot(t'-t)^k.$$

Proof. Consider jobs j_s, $s = r+1,\ldots,\ell$. Since $j_s \in \mathcal{A}_i^2(t)$, $w_{ij_s} \le p_{j_s}(t)\cdot d_{ij}$. Now,

$$k\sum_{s=r+1}^{\ell}\frac{p_{j_s}(t)}{1+\varepsilon}R_{j_s}(t)^{k-1}$$
$$= k\sum_{s=r+1}^{\ell}\frac{p_{j_s}(t)}{1+\varepsilon}\left(R_{j_{s-1}}(t)+p_{j_s}(t)\right)^{k-1}$$
$$\le k2^k\sum_{s=r+1}^{\ell}\frac{p_{j_s}(t)}{1+\varepsilon}R_{j_{s-1}}(t)^{k-1} + \frac{k2^k}{1+\varepsilon}\sum_{s=r+1}^{\ell}p_{j_s}(t)^k$$
$$\overset{(5.10)}{\le} 2^kR_{j_\ell}(t)^k + \frac{k2^k}{1+\varepsilon}\left(\sum_{s=r+1}^{\ell}p_{j_s}(t)\right)^k$$

Since $\sum_{s=r+1}^{\ell}p_{j_s}(t) = (1+\varepsilon)R_{j_\ell}(t)$, and $R_{j_\ell}(t) \le \frac{16k(t'-t)}{\varepsilon}$, we get the result. ∎

Claim 5.5.

$$k\sum_{s=\ell+1}^{m} w_{ij_s}R_{j_s}(t)^{k-1} \le \left(1+\frac{\varepsilon}{2}\right)k\sum_{s=\ell+1}^{m}w_{ij_s}R_{j_s}(t')^{k-1}.$$

Proof. For such jobs j_s, $R_{j_s}(t) \ge \frac{16k(t'-t)}{\varepsilon}$. During $[t,t']$ machine i can process at most $(1+\varepsilon)(t'-t) \le 2(t'-t)$ volume of jobs. So, $R_{j_s}(t') \ge \left(1-\frac{\varepsilon}{8k}\right)R_{j_s}(t)$. ∎

Combining Claims 5.2, 5.3, 5.4, 5.5, we get the statement of the lemma. ∎

Finally, we have our main theorem.

Theorem 5.1. *Algorithm $\mathcal{A}$ is an $O\left(\frac{k}{\varepsilon^{2+\frac{1}{k}}}\right)$-competitive algorithm for minimizing the weighted ℓ_k-norm of flow-time.*

Proof. Let $\mathcal{O}$ denote the optimal schedule. We have

$$\Omega\left(\frac{\varepsilon^{k+1}F^{\mathcal{B},k}}{\gamma}\right) \overset{\text{Lemma 5.4}}{\le} \frac{\alpha_j}{\gamma} - \frac{\sum_i\int_t\beta_{it}dt}{\gamma}$$
$$\overset{\text{Lemmas 5.3, 5.5}}{\le} 2F^{\mathcal{O},k}.$$

Taking k^{th} root gives the theorem. ∎

6 Weighted Flow-time + Energy on Unrelated Machines

We now consider the problem of minimizing the sum of weighted flow-time and energy. We are also given an energy function which has the form $f(s) = s^\gamma$, where $\gamma \ge 1$ is a constant parameter. In other words, if a machine runs at speed s, then the energy consumed per unit time is s^γ. We define $\varepsilon = \frac{1}{\gamma}$. We again need a notion of *speed augmentation*. Let $\tilde{f}$ denote the function

$$\tilde{f}(s) = f\left(s(1-\varepsilon)\right).$$

Note that $\tilde{f}$ allows us to incur the same energy cost as f, however with an extra speed-up of $\frac{1}{1-\varepsilon}$. We shall allow our algorithm to use $\tilde{f}$, whereas the optimal offline algorithm will use f. Note that the notion of $\tilde{f}$ is just for the purpose of designing and analyzing the algorithm. The algorithm will of course incur the cost as dictated by f.

For technical reasons, which will become clear later, it will be easier for us to work with *fractional* weighted flow-time of jobs. Let $p_j(t)$ be the remaining processing time of job j at time t (in a schedule). The fractional remaining weight of the job at time t is defined as $w_j(t) = \frac{p_j(t)}{p_j} w_{ij} = d_{ij} p_j(t)$, where d_j is the density of job j and i is the machine to which it gets dispatched. The fractional weighted flow-time of job j is defined as

$$\int_{t \geq r_j} w_j(t) dt.$$

Our algorithm will be designed for minimizing the sum of total fractional weighted flow-time of jobs and the energy consumed. It will follow from standard arguments that one can derive a competitive algorithm (with an extra γ-factor in the competitive ratio) for minimizing the sum of total weighted flow-time of jobs and the energy consumed.

6.1 The algorithm

In this section, we describe the scheduling algorithm. $\mathcal{A}$. For a machine i and time t, recall that $\mathcal{A}_i(t)$ denotes the set of waiting jobs at time t in the queue of machine i. Let $W_i(t)$ denote the total fractional remaining weight of the jobs in $\mathcal{A}_i(t)$. At time t, a machine i will run at the speed s_t which satisfies $\tilde{f}(s_t) = W_i(t)$. The machine follows HDF (Highest Density First) policy : at any time, it processes the job in its queue with highest density. When a job j, arrives we compute for each machine i the increase in the fractional flow-time if we dispatch j to i – call this quantity Q_{ij}. The job is dispatched to the machine for which Q_{ij} is smallest.

Note that the total fractional weighted flow-time of our algorithm is equal to the total energy consumed (if we use $\tilde{f}$). This algorithm is essentially the same as that of Bansal et al. [5].

6.2 Analysis

We first give some intuition behind the analysis. One can write a convex program for this problem. The idea is again to set the variables in the Lagrangian dual. Recall one crucial (and easy to see) property we used while analyzing the algorithms in previous section : if after some time t, we do not dispatch any jobs to a machine i, then the total weight of remaining jobs at a time $t' \geq t$ on machine i can only decrease as compared to the case when more jobs get dispatched to i after time t. This monotonicity property may not hold here because as more jobs get dispatched to a machine, its speed also increases. In fact, it turns out that such a property may not hold if we are working with the HRDF policy and look at the total weight of remaining jobs. However, if we work with the HDF scheduling policy and only look at the total remaining fractional weight of jobs, then such a property holds. We first show this result. Then we analyze the scheduling algorithm as done in previous sections. The fact that we are now working with a convex rather than a linear program makes the calculations more non-trivial, but the definition of dual variables remains the same.

In the following discussion, we shall assume, for ease of notation, that there is only one machine. We can then apply the result to any of the machines in our input instance.

Given an input instance (with only one machine), we shall define several other instances, which we shall denote by $\mathcal{I}$ or $\mathcal{I}'$. We shall apply our algorithm $\mathcal{A}$ on each of these instances. For an instance $\mathcal{I}$, we shall let $\mathcal{A}^{\mathcal{I}}(t)$ to be the set of jobs waiting at time t in $\mathcal{I}$ (when we run $\mathcal{A}$ on $\mathcal{I}$). Define $w_j^{\mathcal{I}}(t), p_j^{\mathcal{I}}(t)$ similarly. We shall use $W^{\mathcal{I}}(t)$ to denote the total fractional weight of jobs in $\mathcal{A}^{\mathcal{I}}(t)$, i.e., $\sum_{j \in \mathcal{A}^{\mathcal{I}}(t)} w_j^{\mathcal{I}}(t)$. If we do not put a superscript instance $\mathcal{I}$, then we will be refering to the actual input instance. Suppose in an instance $\mathcal{I}$, no job gets released after a certain time t. For a value W, $0 \leq W \leq W^{\mathcal{I}}(t)$, we define the job $j^{\mathcal{I}}(W, t)$ as follows : run the algorithm $\mathcal{A}$ on $\mathcal{I}$, and consider the time $t' \geq t$ when $W^{\mathcal{I}}(t') = W$. Then $j^{\mathcal{I}}(W, t)$ is the job which is being processed at time t'.

A monotonicity property : Consider two instances $\mathcal{I}_1$ and $\mathcal{I}_2$ which are identical except that there is a job j in $\mathcal{I}_2$ which does not appear in $\mathcal{I}_1$. Further, assume that no jobs are released after time r_j in either of the instances.

Claim 6.1. *For all W, $0 \leq W \leq W^{\mathcal{I}_1}(r_j)$,*

$$d_{j^{\mathcal{I}_2}(W, r_j)} \leq d_{j^{\mathcal{I}_1}(W, r_j)}$$

Proof. The set of jobs $\mathcal{A}^{\mathcal{I}_1}(r_j)$ and $\mathcal{A}^{\mathcal{I}_2}(r_j)$ are identical except for the job j which appears in the latter set. Arrange the jobs in $\mathcal{A}^{\mathcal{I}_1}(r_j)$ by decreasing density. Let this ordering be $j_1, \ldots, j_r$. The corresponding order for $\mathcal{A}^{\mathcal{I}_2}(r_j)$ will be same but with j inserted somewhere in the sequence. Since the two instances are identical till r_j, $w_{j_l}^{\mathcal{I}_1}(r_j) = w_{j_l}^{\mathcal{I}_2}(r_j)$ for all l. Now, $j^{\mathcal{I}_1}(W, r_j)$ is the job j_l, where the index l satisfies :

$$w_{j_l}^{\mathcal{I}_1}(r_j) + \ldots + w_{j_r}^{\mathcal{I}_1}(r_j) \leq W < w_{j_{l+1}}^{\mathcal{I}_1}(r_j) + \ldots + w_{j_r}^{\mathcal{I}_1}(r_j).$$

Since the corresponding sequence in $\mathcal{I}_2$ will contain j as well, the job $j^{\mathcal{I}_2}(W, r_j)$ would be a job which comes after j_l (it could be j_l as well) in the sequence. This proves the inequality. ∎

Corollary 6.1. *For any time $t' \geq r_j$,*

$$W^{\mathcal{I}_1}(t') \leq W^{\mathcal{I}_2}(t')$$

Proof. Suppose the inequality is not true – let T be the set of time t' which violate this inequality. Let t^* be the infimum of T. Since, $W^{\mathcal{I}_1}(t'), W^{\mathcal{I}_2}(t')$ are continuous with t', we must have $W^{\mathcal{I}_1}(t^*) = W^{\mathcal{I}_2}(t^*)$ – call this quantity W. Thus, the speed at time t^* is the same in both the schedules. But Claim 6.1 shows that density of $j^{\mathcal{I}_2}(W, t^*)$ is at most the density of $j^{\mathcal{I}_1}(W, t^*)$. Thus, we are processing a job of higher density at time t^* in $\mathcal{I}_1$ than that in $\mathcal{I}_2$. So, for arbitrary small $\varepsilon > 0$, $W^{\mathcal{I}_1}(t^* + \varepsilon) \leq W^{\mathcal{I}_2}(t^* + \varepsilon)$. This contradicts the definition of t^*. ∎

We now prove the monotonicity property. Consider two instances $\mathcal{I}$ and $\mathcal{I}'$ with the following properties. There is a time t such that till t, both instances are identical. In $\mathcal{I}$, no job gets released after t, whereas $\mathcal{I}'$ may see some more jobs after time t (including t). We would like to prove that $W^{\mathcal{I}}(t') \leq W^{\mathcal{I}'}(t')$ for all time $t' \geq t$. Note that this is not entirely obvious because addition of new jobs will speed up the machines, and so jobs will get processed faster.

Lemma 6.1. *Let $\mathcal{I}, \mathcal{I}'$ and time t be as above. Then for any time $t' \geq t$, $W^{\mathcal{I}}(t') \leq W^{\mathcal{I}'}(t')$.*

Proof. The proof is by induction on the number of jobs in $\mathcal{I}'$ which get released during $[t, t']$. If there are no such jobs, then $\mathcal{I}$ and $\mathcal{I}'$ are identical, and so we are done.

Now, let j be the last job which is released in $\mathcal{I}'$ during $(t, t']$. Let $\mathcal{I}''$ be the instance which is same as $\mathcal{I}'$ except for the job j. By induction hypothesis, $W^{\mathcal{I}}(t') \leq W^{\mathcal{I}''}(t')$. So we just need to compare $W^{\mathcal{I}''}(t')$ and $W^{\mathcal{I}'}(t')$. But this follows from Corollary 6.1, where $\mathcal{I}_1 = \mathcal{I}'', \mathcal{I}_2 = \mathcal{I}'$. ∎

The Convex Program and its Lagrangian Dual : We first write a convex programming relaxation. For a job j, let s_{ijt} be the speed with which we are processing it at time t on machine i. The convex program is given in Figure 3. We show that it is indeed a relaxation.

Lemma 6.2. *Consider a non-migratory schedule $\mathcal{S}$ for an instance $\mathcal{I}$ of this problem. Let s_{ijt} be the corresponding solution to this convex program. Then the objective value of the solution s_{ijt} for this convex program is at most a constant times the sum of the weighted fractional flow-time and the energy consumed by $\mathcal{S}$.*

Proof. The constraint 6.25 is stating that a job j must be processed to full extent. So, s_{ijt} must satisfy this constraint. Let us look at the objective value. For a job j, its fractional weighted flow-time of j can be written as

$$\begin{aligned}\sum_i \int_{t \geq r_j} d_{ij} p_j(t) dt &= d_{ij} \int_{t \geq r_j} \left(\int_{t' \geq t} s_{ijt'} dt' \right) dt \\ &= \int_{t' \geq r_j} d_{ij} s_{ijt'} (t' - r_j) dt.\end{aligned}$$

Thus, the first term in the objective function is capturing the weighted fractional flow-time, and the second term represents the total energy consumed. Let us now show that the third term is at most a constant times the cost of $\mathcal{S}$.

Suppose $\mathcal{S}$ schedules a job j on machine i. Let T be the time by which i processes half of j, i.e., does $p_{ij}/2$ amount of processing on j. Let s be the *average* speed of i while processing j during $[r_j, T]$, i.e., $s = \frac{1}{T - r_j} \cdot \int_{r_j}^{T} s_{ijt} dt = \frac{p_{ij}}{2(T - r_j)}$. Since the energy function is a convex function, it is easy to check that the energy consumed while processing j is at least $(T - r_j) s^\gamma = \frac{p_{ij}}{2s} s^\gamma$. Also, the total energy consumed by $\mathcal{S}$ is at least the sum over all jobs j of the energy consumed while processing j only. Further, the fractional weighted flow-time of j is at least $(T - r_j) \cdot \frac{w_{ij}}{2} = \frac{w_{ij} p_{ij}}{4s}$. Now observe that

$$p_{ij} \cdot \left(\frac{s^{\gamma - 1}}{2} + \frac{w_{ij}}{4s} \right) \geq \frac{1}{4} w_{ij}^{1-\varepsilon} p_{ij}.$$

So, we see that the cost of $\mathcal{S}$ is at least $\frac{1}{4} \sum_j w_{i(j)j}^{1-\varepsilon} p_{i(j)j}$, where $i(j)$ is the machine on which j gets dispatched by $\mathcal{S}$. Now the third term in the objective function is

$$\begin{aligned}\sum_{j,i} w_{ij}^{1-\varepsilon} \int_{t \geq r_j} s_{ijt} dt &= \sum_j w_{i(j)j}^{1-\varepsilon} \int_{t \geq r_j} s_{i(j)jt} dt \\ &= \sum_j w_{i(j)j}^{1-\varepsilon} p_{ij}.\end{aligned}$$

This proves the lemma. ∎

The Lagrangian dual where we have Lagrangian variables for the constraints (6.25), is shown in Figure 3 (the s_{ijt} variables are defined only for $t \geq r_j$).

Setting the Dual Variables We now show how to set the α_j variables. Fix a job j. We define α_j as the increase in weighted fractional flow-time assuming no new job arrives after j, i.e., $\alpha_j = \min_i Q_{ij}$, where Q_{ij} is the increase in weighted fractional flow-time if we dispatch j to machine i. Our result will follow from the following theorem.

Primal Relaxation

$$\min \sum_j \int_{t \geq r_j} s_{jt} d_j(t - r_j)dt + \int_t f\left(\sum_j s_{jt}\right) dt$$

$$+ \sum_{j,i} w_{ij}^{1-\varepsilon} \int_{t \geq r_j} s_{ijt} dt \tag{6.24}$$

$$\sum_i \int_{t \geq r_j} \frac{s_{ijt}}{p_{ij}} dt = 1 \qquad \forall j \tag{6.25}$$

$$s_{ijt} \geq 0 \quad \forall i, j, t \geq r_j \tag{6.26}$$

Lagrangian Dual

$$\max_{\alpha_j} \sum_j \alpha_j - \sum_i \int_t \max_{s_{ijt}} \Bigg[\sum_j s_{ijt}\Big(\frac{\alpha_j}{p_{ij}}$$

$$-d_{ij}(t - r_j) - w_{ij}^{1-\varepsilon}\Big) - f\left(\sum_j s_{ijt}\right) \Bigg] dt \tag{6.27}$$

$$s_{ijt} \geq 0 \quad \forall i, j, t \tag{6.28}$$

Figure 3: The convex programming relaxation and its Lagrangian dual

Theorem 6.1. *Fix a time t' and machine i. Then, the maximum value of*

$$\sum_j s_{ijt'} \left(\frac{\alpha_j}{p_{ij}} - d_{ij}(t' - r_j) - w_{ij}^{1-\varepsilon} \right) - f\left(\sum_j s_{ijt'}\right), \tag{6.29}$$

where the maximum is taken over $s_{ijt'}$ for all jobs j, $s_{ijt'} \geq 0$, is at most $W_i(t')(1-\varepsilon)$.

Before we prove this theorem, let us see why it implies the desired result.

Corollary 6.2. *The algorithm $\mathcal{A}$ is $O(\gamma)$-competitive for the objective of minimizing the sum of weighted fractional flow-time and energy. This yields an $O(\gamma^2)$-competitive algorithm for the objective of minimizing the sum of weighted flow-time and energy.*

Proof. Let $F^{\mathcal{A}}$ be the total weighted fractional flow-time of the algorithm $\mathcal{A}$. Then the theorem above shows that the value of the Lagrangian relaxation is at least $\sum_j \alpha_j - (1-\varepsilon)F^{\mathcal{A}}$. But, $\sum_j \alpha_j$ is equal to $F^{\mathcal{A}}$, and hence, the value of the Lagrangian relaxation is at least $\varepsilon F^{\mathcal{A}}$. But, we know that the optimal value of the Lagrangian dual is $O(\texttt{opt})$, where $\texttt{opt}$ denotes the value of the optimal solution (Lemma 6.2). Thus, $F^{\mathcal{A}}$ is at most $O(\texttt{opt}/\varepsilon)$. Now, we need to compute the energy incurred by $\mathcal{A}$. If we assume that the energy function is $\tilde{f}$, then the energy consumed is same as $F^{\mathcal{A}}$. This is so because at any point of time t, we choose the speed s such that $\tilde{f}(s) = W(t)$. However, the energy function is f, and so we pay an extra factor of $\left(\frac{1}{1-\varepsilon}\right)^{\gamma}$, which is only a constant. This proves the first statement.

For the second statement, we run the algorithm $\mathcal{A}$, but now we speed up each machine by a factor of $(1+\varepsilon)$. Call this schedule $\mathcal{B}$. Using standard arguments (see e.g., [6]) the total weighted flow-time of $\mathcal{B}$ is at most $O(\gamma)$ times the total weighted fractional flow-time of $\mathcal{A}$. Further, the energy consumed by $\mathcal{B}$ is only a constant factor more than that by $\mathcal{A}$. ■

We now prove Theorem 6.1. Fix a time t' and machine i for the rest of this discussion. Consider the values $s_{ijt'}$ which maximize the quantity (6.29). We first observe that exactly one of these values will be non-zero. Indeed, suppose there are two jobs j_1 and j_2 such that $s_{ij_1t'}, s_{ij_2t'} > 0$. Then either the values $s_{ij_1t'} + \delta, s_{ij_2t'} - \delta$, or $s_{ij_1t'} - \delta, s_{ij_2t'} + \delta$ will improve the quantity in (6.29). Thus, we will be done if we prove the following lemma.

Lemma 6.3. *For any job $j, r_j \leq t'$, and $s \geq 0$,*

$$s\left(\frac{\alpha_j}{p_{ij}} - d_{ij}(t' - r_j) - w_{ij}^{1-\varepsilon}\right) - f(s) \leq W_i(t')(1-\varepsilon).$$

Proof. Fix a job j which gets released at time t. Rearranging terms, it is enough to show that for all s,

$$\frac{\alpha_j}{p_{ij}} - d_{ij}(t' - t) - w_{ij}^{1-\varepsilon} \leq s^{\gamma-1} + \frac{W_i(t')(1-\varepsilon)}{s}.$$

We would like to find the value of s which minimizes the right-hand side. By using differentiation, one can easily check that the right hand side above is at least $W_i(t')^{1-\varepsilon}$. Thus, we will be done if we show that

$$\frac{\alpha_j}{p_{ij}} - d_j(t' - t) - w_{ij}^{1-\varepsilon} \leq W_i(t')^{1-\varepsilon}$$

Using Lemma 6.1, we can assume that no jobs arrive after time t (because this will not change the left hand side above, and will only increase the right hand side). At time t, let the jobs in $\mathcal{A}_i(t)$ arranged by descending order of density be $j_1, \dots, j_m$, and suppose the density of j lies between those of j_k and j_{k+1}. Assume that j does not get dispatched to i – the other case is similar. Let J_1 denote the set of jobs $\{j_1, \dots, j_k\}$ and J_2 be the jobs in $\{j_{k+1}, \dots, j_m\}$. Let $W_i^1(t)$ be the total fractional remaining weight of jobs in J_1 at time t, i.e., $\sum_{l=1}^k w_{j_l}(t)$. Define $W_i^2(t)$ for J_2 similarly. Suppose it took x_j units of time to process job j if j were dispatched to i (note that j would be processed after J_1 and before J_2). Further let T_0 be the time at which the jobs in J_1 finish processing on machine i. Then we can bound α_j by

$$\alpha_j \le w_{ij}(T_0 - t) + w_{ij}x_j + W_i^2(t)x_j. \tag{6.30}$$

Indeed, because of the arrival of j, the fractional flow-time of jobs in J_1 will decrease because the machines will now run at faster speed. So the increase in fractional weighted flow-time can be bounded by the weighted flow-time of j plus the increase in that of jobs in J_2. The latter is at most $x_j \cdot W_i^2(t)$ because the delay in the completion of a job in J_2 is at most x_j (note that the jobs in J_1 will finish earlier). We now consider two cases depending on the time t'.

First assume that $t' \le T_0$. So, the machine i is processing a job in J_1 at time t'.

Claim 6.2.

$$T_0 - t' \le \frac{1}{d_{ij}}\big(W_i(t')^{1-\varepsilon} - W_i^2(t)^{1-\varepsilon}\big).$$

Proof. Fix a time u between t' and T_0. Let $W_i(u)$ be the total remaining fractional weight at time u. Then, the speed of machine i at this time is $\frac{W_i(u)^\varepsilon}{1-\varepsilon}$. So, in a small amount of time du, we process $\frac{W_i(u)^\varepsilon}{1-\varepsilon}du$ volume of jobs. Since the density of these jobs is at least d_{ij}, the decrease in the total fractional remaining weight, dW is at least

$$\frac{d_{ij}W_i(u)^\varepsilon}{(1-\varepsilon)}du.$$

So we can set up the differential equation

$$\int_{W_i^2(t)}^{W_i(t')} \frac{(1-\varepsilon)dW}{d_{ij}W^{\frac{1}{\gamma}}} \ge \int_{t'}^{T_0} du.$$

Integrating this gives the claim. ∎

Corollary 6.3. *x_j is at most* $\min\left(p_{ij}w_{ij}^{-\varepsilon}, \frac{p_{ij}(1-\varepsilon)}{W_i^2(t)^\varepsilon}\right)$.

Proof. If j were dispatched to machine i, we can show by the same proof as that of Claim 6.2 that the machine i will process j for $\frac{1}{d_{ij}}\big((W_i^2(t) + w_{ij})^{1-\varepsilon} - W_i^2(t)^{1-\varepsilon}\big)$ amount of time. But the latter is at most $w_{ij}^{1-\varepsilon}$. This proves the first inequality.

The second inequality follows because while the job j is getting processed, the remaining fractional weight is at least $W_i^2(t)$. So the speed will be at least $W_i^2(t)^\varepsilon$. ∎

Now, Corollary 6.3 and inequality (6.30) imply that

$$\begin{aligned}
\frac{\alpha_j}{p_{ij}} &\le d_{ij}(T_0 - t) + d_{ij}p_{ij}w_{ij}^{-\varepsilon} + \frac{x_j W_i^2(t)}{p_{ij}} \\
&\le d_{ij}(T_0 - t) + w_{ij}^{1-\varepsilon} + W_i^2(t)^{1-\varepsilon}(1-\varepsilon) \\
&= d_{ij}(T_0 - t') + w_{ij}^{1-\varepsilon} + d_{ij}(t' - t) \\
&\quad + W_i^2(t)^{1-\varepsilon}(1-\varepsilon) \\
&\overset{\text{Claim 6.2}}{\le} d_{ij}(t' - t) + w_{ij}^{1-\varepsilon} + W_i(t')^{1-\varepsilon}
\end{aligned}$$

This implies the lemma. Now, we consider the case when $t' > T_0$, i.e., the machine i is processing a job of J_2 at time t. So, $W(t')$ is same as $W_i^2(t')$.

Claim 6.3.

$$t' - T_0 \ge \frac{1}{d_{ij}}\big(W_i^2(t)^{1-\varepsilon} - W_i(t')^{1-\varepsilon}\big)$$

Proof. Consider a time u between T_0 and t'. Let $W_i(u)$ be the remaining fractional weight at time u. Then, as in the proof of Claim 6.2, the total processing done during $[u, u+du]$ is $\frac{W_i(u)^\varepsilon}{1-\varepsilon}du$. Since the density of jobs in J_2 is at most d_{ij}, the total fractional weight dW processed during this time is at most $\frac{d_{ij}W_i(u)^\varepsilon}{1-\varepsilon}du$. Thus we get the differential equation :

$$\int_{W_i(t')}^{W_i^2(t)} \frac{(1-\varepsilon)dW}{d_{ij}W^\varepsilon} \le \int_{T_0}^{t'} du.$$

Integrating this gives the result. ∎

Again, using Corollary 6.3 and inequality (6.30), we get

$$\begin{aligned}
\frac{\alpha_j}{p_{ij}} &\le d_{ij}(T_0 - t) + d_{ij}p_{ij}w_{ij}^{-\varepsilon} + \frac{W_i^2(t')x_j}{p_{ij}} \\
&\le d_{ij}(t' - t) - d_{ij}(t' - T_0) + w_{ij}^{1-\varepsilon} \\
&\quad + (1-\varepsilon)W_i^2(t')^{1-\varepsilon} \\
&\overset{\text{Claim 6.3}}{\le} d_{ij}(t' - t) + w_{ij}^{1-\varepsilon} + W_i^2(t')^{1-\varepsilon}
\end{aligned}$$

This implies the result. ∎

References

[1] N. Avrahami and Y. Azar. Minimizing total flow time and total completion time with immediate dispatching. In *SPAA*, pages 11–18. ACM, 2003.

[2] N. Bansal and K. Pruhs. Server scheduling in the ℓ_p norm: A rising tide lifts all boats. In *ACM Symposium on Theory of Computing*, pages 242–250, 2003.

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

[4] Nikhil Bansal and Kirk Pruhs. Server scheduling in the weighted lp norm. In *LATIN*, pages 434–443, 2004.

[5] Nikhil Bansal, Kirk Pruhs, and Clifford Stein. Speed scaling for weighted flow time. *SIAM J. Comput.*, 39(4):1294–1308, 2009.

[6] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In *STOC*, pages 679–684, 2009.

[7] C. Chekuri, S. Khanna, and A. Zhu. Algorithms for weighted flow time. In *STOC*, pages 84–93. ACM, 2001.

[8] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In *STOC*, pages 363–372, 2004.

[9] Naveen Garg and Amit Kumar. Better algorithms for minimizing average flow-time on related machines. In *ICALP (1)*, pages 181–190, 2006.

[10] Naveen Garg and Amit Kumar. Minimizing average flow-time : Upper and lower bounds. In *FOCS*, pages 603–613, 2007.

[11] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Scalably scheduling power-heterogeneous processors. In *ICALP (1)*, pages 312–323, 2010.

[12] Sungjin Im and Benjamin Moseley. An online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines. In *SODA*, 2011.

[13] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. In *IEEE Symposium on Foundations of Computer Science*, pages 214–221, 1995.

[14] Hans Kellerer, Thomas Tautenhahn, and Gerhard J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. In *STOC*, pages 418–426, 1996.