Online Non-clairvoyant Scheduling to Simultaneously Minimize All Convex Functions

Kyle Fox[1]⋆, Sungjin Im[2], Janardhan Kulkarni[2]⋆⋆, and Benjamin Moseley[3]

[1] Department of Computer Science, University of Illinois, Urbana, IL 61801. kylefox2@illinois.edu

[2] Department of Computer Science, Duke University, Durham NC 27708. [sungjin,kulkarni]@cs.duke.edu

[3] Toyota Technological Institute at Chicago, Chicago, IL 60637. moseley@ttic.edu

Abstract. We consider scheduling jobs online to minimize the objective $\sum_{i\in[n]} w_i g(C_i - r_i)$, where w_i is the weight of job i, r_i is its release time, C_i is its completion time and g is any non-decreasing convex function. Previously, it was known that the clairvoyant algorithm Highest-Density-First (HDF) is $(2+\epsilon)$-speed $O(1)$-competitive for this objective on a single machine for any fixed $0 < \epsilon < 1$ [21]. We show the first non-trivial results for this problem when g is not concave and the algorithm must be *non-clairvoyant*. More specifically, our results include:

- A $(2+\epsilon)$-speed $O(1)$-competitive non-clairovyant algorithm on a single machine for all non-decreasing convex g, matching the performance of HDF for any fixed $0 < \epsilon < 1$.
- A $(3+\epsilon)$-speed $O(1)$-competitive non-clairovyant algorithm on multiple identical machines for all non-decreasing convex g for any fixed $0 < \epsilon < 1$.

Our positive result on multiple machines is the first non-trivial one even when the algorithm is clairvoyant. Interestingly, all performance guarantees above hold for all non-decreasing convex functions g *simultaneously*. We supplement our positive results by showing any algorithm that is oblivious to g is not $O(1)$-competitive with speed less than 2 on a single machine. Further, any non-clairvoyent algorithm that knows the function g cannot be $O(1)$-competitive with speed less than $\sqrt{2}$ on a single machine or speed less than $2 - \frac{1}{m}$ on m identical machines.

1 Introduction

Scheduling a set of jobs that arrive over time on a single machine is perhaps the most basic setting considered in scheduling theory. A considerable amount of work has focused on this fundamental problem. For examples, see [26]. In

⋆ Research by this author is supported in part by the Department of Energy Office of Science Graduate Fellowship Program (DOE SCGF), made possible in part by the American Recovery and Reinvestment Act of 2009, administered by ORISE-ORAU under contract no. DE-AC05-06OR23100.

⋆⋆ Supported by NSF awards CCF-1008065 and IIS-0964560.

this setting, there are n jobs that arrive over time, and each job i requires some processing time p_i to be completed on the machine. In the *online* setting, the scheduler becomes first aware of job i at time r_i when job i is released. Note that in the online setting, it is standard to assume jobs can be *preempted*.

Generally, a client that submits a job i would like to minimize the *flow time* of the job defined as $F_i := C_i - r_i$, where C_i denotes the completion time of job i. The flow time of a job measures the amount of time the job waits to be satisfied in the system. When there are multiple jobs competing for service, the scheduler needs to make scheduling decisions to optimize a certain global objective. One of the most popular objectives is to minimize the total (or equivalently average) flow time of all the jobs, i.e., $\sum_{i \in [n]} F_i$. It is well known that the algorithm Shortest-Remaining-Processing-Time (SRPT) is optimal for that objective in the single machine setting. The algorithm SRPT always schedules the job that has the shortest remaining processing time at each point in time. Another well known result is that the algorithm First-In-First-Out (FIFO) is optimal for minimizing the maximum flow time, i.e., $\max_{i \in [n]} F_i$ on a single machine. The algorithm FIFO schedules the jobs in the order they arrive.

These classic results have been extended to the case where jobs have priorities. In this extension, each job i is associated with a weight w_i denoting its priority; large weight implies higher priority. The generalization of the total flow time problem is to minimize the total weighted flow time, $\sum_{i \in [n]} w_i F_i$. For this problem it is known that no online algorithm can be $O(1)$-competitive [5]. A generalization of the maximum flow time problem is to minimize the maximum weighted flow time $\max_{i \in [n]} w_i F_i$. It is also known for this problem that no online algorithm can be $O(1)$-competitive [11,15].

Due to these strong lower bounds, previous work for these objectives has appealed to the relaxed analysis model called resource augmentation [22]. In this relaxation, an algorithm A is said to be s-speed c-competitive if A has a competitive ratio of c when processing jobs s times faster than the adversary. The primary goal of a resource augmentation analysis is to find the minimum speed an algorithm requires to be $O(1)$-competitive. For the total weighted flow time objective, it is known that the algorithm Highest-Density-First (HDF) is $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for any fixed $\epsilon > 0$ [25,10]. The algorithm HDF always schedules the job i of highest density, $\frac{w_i}{p_i}$. For the maximum weighted flow objective, the algorithm Biggest-Weight-First (BFW) is known to be $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive [15]. BFW always schedules the job with the largest weight.

Another widely considered objective is minimizing the ℓ_k-norms of flow time, $\left(\sum_{i \in [n]} F_i^k\right)^{1/k}$ [8,17,20,1,4,23]. The ℓ_k-norm objective is most useful for $k \in \{1, 2, 3, \infty\}$. Observe that total flow time is the ℓ_1-norm of flow time, and the maximum flow time is the ℓ_∞-norm. The ℓ_2 and ℓ_3 norms are natural balances between the ℓ_1 and ℓ_∞ norms. These objectives can be used to decrease the variance of flow time, thereby yielding a schedule that is fair to requests. It is known that no algorithm can be $n^{\Omega(1)}$-competitive for minimizing the ℓ_2-norm

[8]. On the positive side, for $\epsilon > 0$, HDF was shown to be $(1+\epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive for any ℓ_k-norm objective, $k \geq 1$ [8].

These objectives have also been considered in the identical machine scheduling setting [24,14,3,2,9,16,13,19]. In this setting, there are m machines that the jobs can be scheduled on. Each job can be scheduled on any machine and job i requires processing time p_i no matter which machine it is assigned to. In the identical machine setting it is known that any randomized online algorithm has competitive ratio $\Omega(\min\{\frac{n}{m}, \log P\})$, where P denotes the ratio between the maximum and minimum processing time of a job [24]. HDF as well as several other algorithms are known to be scalable for weighted flow time [10,14,19,13]. For the ℓ_k-norms objective the multiple machine version of HDF is known to be scalable [13] as well as other algorithms [14,19]. For the maximum unweighted flow it is known that FIFO is $(3-2/m)$-competitive, and for weighted maximum flow time a scalable algorithm is known [11,15].

The algorithms HDF and SRPT use the processing time of a job to make scheduling decisions. An algorithm which learns the processing time of a job upon its arrival is called *clairvoyant*. An algorithm that does not know the processing time of a job before completing the job is said to be *non-clairvoyant*. Among the aforementioned algorithms, FIFO and BFW are non-clairvoyant. Non-clairvoyant schedulers are highly desirable in many real world settings. For example, an operating system typically does not know a job's processing time. Thus, there has been extensive work done on designing non-clairvoyant schedulers for the problems discussed above. Scalable non-clairvoyant algorithms are known for the maximum weighted flow time, average weighted flow time, and ℓ_k-norms of flow time objectives even on identical machines [15,14].

It is common in scheduling theory that algorithms are tailored for specific scheduling settings and objective functions. For instance, FIFO is considered the best algorithm for non-clairvoyantly minimizing the maximum flow time, while HDF is considered one of the best algorithms for minimizing total weighted flow time. One natural question that arises is what to do if a system designer wants to minimize several objective functions simultaneously. For instance, a system designer may want to optimize average quality of service, while minimizing the maximum waiting time of a job. Different algorithms have been considered for minimizing average flow time and maximum flow time, but the system designer would like to have a single algorithm that performs well for both objectives.

Motivated by this question, the general cost function objective was considered in [21]. In the general cost function problem, there is a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ given, and the goal of the scheduler is to minimize $\sum_{i \in [n]} w_i g(F_i)$. One can think of $g(F_i)$ as the penalty of making job i wait F_i time steps, scaled by job i's priority (its weight w_i). This objective captures most scheduling metrics. For example, this objective function captures total weighted flow time by setting $g(x) = x$. By setting $g(x) = x^k$, the objective also captures minimizing $\sum_{i \in [n]} F_i^k$ which is essentially the same as the ℓ_k-norm objective except the outer kth root is not taken. Finally, by making g grow very quickly the objective can be designed to capture minimizing the maximum weighted flow time. As stated, one of the reasons this objective was introduced was to find an algorithm that can optimize several objectives simultaneously. If one were to design an algorithm that optimizes the general cost function g while being oblivious to g, then this algorithm would optimize *all* objective functions in this framework *simultaneously*.

In [21], the general cost function objective was considered only assuming that g is non-decreasing. This is a natural assumption since there should be no incentive for a job to wait longer. It was shown that in this case, no algorithm that is oblivious to the cost function g can be $O(1)$-competitive with speed $2 - \epsilon$ for any fixed $\epsilon > 0$. Surprisingly, it was also shown that HDF, an algorithm that is oblivious to g, is $(2 + \epsilon)$-speed $O(1/\epsilon)$-competitive. This result shows that it is indeed possible to design an algorithm that optimizes most of the reasonable scheduling objectives simultaneously on a single machine. Recall that HDF is clairvoyant. Ideally, we would like to have a non-clairvoyant algorithm for general cost functions. Further, there is currently no known similar result in the multiple identical machines setting.

Results: In this paper, we consider non-clairvoyant online scheduling to minimize the general cost function on a single machine as well as on multiple identical machines. In both the settings, we give the *first* nontrivial positive results when the online scheduler is required to be non-clairvoyant. We concentrate on cost functions g which are differentiable, non-decreasing, and convex. We assume without loss of generality that $g(0) = 0$. Note that all of the objectives discussed previously have these properties. We show the following somewhat surprising result (Section 4).

Theorem 1. *There exists a non-clairvoyant algorithm that is $(2+\epsilon)$-speed $O(1/\epsilon)$-competitive for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on a single machine for any $\epsilon > 0$, when the given cost function $g : \mathbb{R}^+ \to \mathbb{R}^+$ is differentiable, non-decreasing, and convex (g' is non-decreasing). Further, this algorithm is oblivious to g.*

We then consider the general cost function objective on multiple machines for the first time, and give a positive result. This algorithm is also non-clairvoyant.

Theorem 2. *There exists a non-clairvoyant algorithm that is $(3+\epsilon)$-speed $O(1/\epsilon)$-competitive for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on multiple identical machines for any $\epsilon > 0$, when the given cost function $g : \mathbb{R}^+ \to \mathbb{R}^+$ is differentiable, non-decreasing, and convex (g' is non-decreasing). Further, this algorithm is oblivious to g.*

Note that we do not know if there exists a constant competitive non-clairvoyant algorithm even for a single machine with any constant speed when the cost function is neither convex nor concave. We leave this gap as an open problem.

We complement these positive results by extending the lower bound presented in [21]. They showed that for any $\epsilon > 0$, no oblivious algorithm can be $(2 - \epsilon)$-speed $O(1)$-competitive on a single machine when the cost function g is non-decreasing, but perhaps discontinuous. We show the same lower bound even if g is differentiable, non-decreasing, and convex. Thus, on a single machine, our positive result is essentially tight up to constant factors in the competitive ratio, and our algorithm achieves the same performance guarantee while being non-clairvoyant.

Theorem 3. *No randomized clairvoyant algorithm that is oblivious to g can be $(2-\epsilon)$-speed $O(1)$-competitive for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on a single machine even if all jobs have unit weights and g is differentiable, non-decreasing, and convex.*

We go on to show that even if a non-clairvoyant algorithm knows the cost function g, the algorithm cannot have a bounded competitive ratio when given speed less than $\sqrt{2}$.

Theorem 4. *Any deterministic non-clairvoyant (possibly aware of g) algorithm for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on a single machine has an unbounded competitive ratio when given speed $\sqrt{2}-\epsilon$ for any fixed $\epsilon > 0$ where g is differentiable, non-decreasing, and convex..*

Finally, we show that at least $2 - \frac{1}{m}$ speed is needed for any non-clairvoyant algorithm to be constant competitive on m identical machines. This is the first lower bound for the general cost function specifically designed for the multiple machine case.

Theorem 5. *Any randomized non-clairvoyant (possibly aware of g) algorithm on m identical machines has an unbounded competitive ratio when given speed less than $2 - \frac{1}{m} - \epsilon$ for any fixed $\epsilon > 0$ when g is differentiable, non-decreasing, and convex..*

Techniques: To show Theorem 1, we consider the well-known algorithm Weighted-Shortest-Elapsed-Time-First (WSETF) on a singe machine, and first show that it is 2-speed $O(1)$-competitive for minimizing the *fractional* version of the general cost function objective. Then with a small extra amount of speed augmentation, we convert WSETF's schedule into the one that is $(2+\epsilon)$-speed $O(1)$-competitive for the integral general cost function. This conversion is now a fairly standard technique, and will be further discussed in Section 2. This conversion was also used in [21] when analyzing HDF. One can think of the fractional objective as converting each job i to a set of p_i unit sized jobs of weight w_i/p_i. That is, the weight of the job is distributed among all unit pieces of the job. Notice that the resulting weight of the unit time jobs as well as the number of them depends on the job's original processing time. Thus, to analyze a non-clairvoyant algorithm for the fractional instance one must consider the algorithm's decisions on the original instance and argue about the algorithm's cost on the fractional instance. This differs from the analysis of [21], where the clairvoyant algorithm HDF can assume full knowledge of the conversion. Due to this, in [21] they can argue directly about HDF's decisions for the fractional instance of the problem. Since a non-clairvoyant algorithm does not know the fractional instance, it seems difficult to adapt the techniques of [21] when analyzing a non-clairvoyant algorithm.

If the instance consists of a set of unweighted jobs, WSETF always processes the job which has been processed the least. Let $q_i^A(t)$ be the amount WSETF has processed job i by time t. When jobs have weights, WSETF processes the job i such that $\frac{w_i}{q_i^A(t)}$ is maximized where w_i is the weight in the integral instance. One can see that WSETF will not necessarily process the jobs with the highest weight at each time, which is what the algorithm HDF will do if all jobs are unit sized. Further, WSETF may round robin among multiple jobs of the same priority. For these reasons, our analysis of WSETF is substantially different from the analysis in [21], and relies crucially on a new lower bound we develop on the optimal solution. This lower bound holds for any objective that is differentiable, non-decreasing, and convex. Our lower bound gives a way to relate the final objective of the optimal solution to the volume of unsatisfied work the optimal solution has at each moment in time. We then bound the volume of unsatisfied jobs in the optimal schedule at each moment in time and relate this to WSETF's instantaneous increase in its objective function. We believe that our new lower bound will be useful in further analysis of scheduling problems since it is versatile enough to be used for many scheduling objectives.

Other Related Work: For minimizing average flow time on a single machine, the non-clairvoyant algorithms Shortest Elapse Time First (SETF) and Latest Arrival Processor Sharing (LAPS) are known to be scalable [22,18]. Their weighted versions Weighted Shortest Elapse Time First (WSETF) and Weighted Latest Arrival Processor Sharing (WLAPS) are scalable for average weighted flow time [8,6], and also for (weighted) ℓ_k norms of flow time [8,17].

In [21], Im et al. showed Weighted Latest Arrival Processor Sharing (WLAPS) is scalable for concave functions g. They also showed that no online randomized algorithm, even with any constant speed-up, can have a constant competitive ratio, when each job i has its own cost function g_i, and the goal is to minimize $\sum_{i\in[n]} g_i(F_i)$. This more general problem was studied in the offline setting by Bansal and Pruhs [7]. They gave an $O(\log\log nP)$-approximation (without speed augmentation), where P is the ratio of the maximum to minimum processing time of a job. This is the best known approximation for minimizing average weighted flow time offline, and a central open question in scheduling theory is whether or not a $O(1)$-approximation exists for weighted flow time offline.

2 Preliminaries

The Fractional Objective: In this section we define the fractional general cost objective and introduce some notation. We will refer to the non-fractional general cost objective as *integral*. For a schedule, let $p_i(t)$ denote the remaining processing time of job i at time t. Let $\beta_i(p)$ be the latest time t such that $p_i(t) = p$ for any p where $0 \leq p \leq p_i$.

The fractional objective penalizes jobs over time by charging in proportion to how much of the job remains to be processed. Formally, the fractional objective is defined as:

$$\sum_{i\in[n]} \int_{t=r_i}^{C_i} \frac{w_i p_i(t)}{p_i} g'(t-r_i)\mathrm{dt} \tag{1}$$

Generally when the fractional objective is considered, it is stated in the form (1). For our analysis it will be useful to note that this objective is equivalent to:

$$\sum_{i\in[n]} \frac{w_i}{p_i} \int_{p=0}^{p_i} g(\beta_i(p)-r_i)\mathrm{dp} \tag{2}$$

As noted earlier, considering the fractional objective has proven to be quite useful for the analysis of algorithms in scheduling theory, because directly arguing about the fractional objective is usually easier from an analysis viewpoint. A schedule which optimizes the fractional objective can then be used to get a good schedule for the integral objective as seen in the following theorems. In the first theorem (6), the algorithm's fractional cost is compared against the optimal solution for the fractional objective. In the second theorem (7), the algorithm's fractional cost is compared against the optimal solution for the integral instance.

Theorem 6 ([21]). *If a (non-clairvoyant) algorithm A is s-speed c-competitive for minimizing the fractional general cost function then there exists a $(1+\epsilon)s$-speed $\frac{(1+\epsilon)c}{\epsilon}$-competitive (non-clairvoyant) algorithm for the integral general cost function objective for any $0 \leq \epsilon \leq 1$.*

Theorem 7 ([21]). *If a (non-clairvoyant) algorithm A with s-speed has fractional cost at most a factor c larger than the optimal solution for the integral objective then there exists a $(1+\epsilon)s$-speed $\frac{(1+\epsilon)c}{\epsilon}$-competitive (non-clairvoyant) algorithm for the integral general cost function objective for any $0 \leq \epsilon \leq 1$.*

These two theorems follow easily by the analysis given in [21]. We note that the resulting algorithm that performs well for the integral objective is not necessarily the algorithm A. Interestingly, [21] shows that if A is HDF then the resulting algorithm is still HDF. However, if A is WSETF, the resulting integral algorithm need not be WSETF.

Notation: We now introduce some more notation that will be used throughout the paper. For a schedule B, let C_i^B be the completion time of job i. Let $p_i^B(t)$ denote the remaining processing time for job i at time t. Let $q_i^B(t) = p_i - p_i^B(t)$ be the amount job i has been processed by time t. Let $p_{i,j}^B(t) = (\min\{\frac{w_j}{w_i}p_i, p_j\} - q_j^B(t))^+$. Here $(\cdot)^+$ denotes $\max\{\cdot, 0\}$. Let $p_{i,j} = \min\{\frac{w_j}{w_i}p_i, p_j\} = p_{i,j}^B(r_j)$. If the schedule B is that produced by WSETF and $t \in [r_i, C_i^B]$ then $p_{i,j}^B(t)$ is exactly the amount of processing time WSETF will devote to job j during the interval $[t, C_i^B]$. In other words, the remaining time job i waits due to WSETF processing job j. Let $Q_B(t)$ be the set of job released but unsatisfied by B at time t. Let

$Z_i^B(t) = \sum_{j \in Q_B(t)} p_{i,j}^B(t)$. When the algorithm B is the optimal solution (OPT) we set B to be O and if the algorithm is WSETF we set B to be A. For example $Q_A(t)$ is the set of released and unsatisfied jobs for WSETF at time t. Finally, for a set of possibly overlapping time intervals I, let $|I|$ denote the total length of their union.

3 Analysis tools

In this section we introduce some useful tools that we use for our analysis. First we present our novel lower bound on the optimal solution. This lower bound is the key to our analysis and the main technical contribution of the paper. The left-hand-side of the inequality in the lemma has an arbitrary function $x(t) : \mathbb{R}^+ \to \mathbb{R}^+ \setminus \{0\}$, while the right-hand-side is simply a fractional cost of the schedule in consideration. This lower bound is inspired by one presented in [20]. However, the lower bound given in [20] involves substantially different terms, and is only for the ℓ_k-norms of flow time. Our proof is considerably different from [20], and perhaps simpler. Since this lower bound applies to any objective that fits into the general cost function framework, we believe that this lower bound will prove to be useful for a variety of scheduling problems. The assumption in the lemma that g is convex is crucial; the lemma is not true otherwise. The usefulness of this lemma will become apparent in the following two sections. We prove this lemma in Section 6 after we show the power of the lemma.

Lemma 1. *Let σ be a set of jobs on a single machine with speed s'. Let B be any feasible schedule and $B(\sigma)$ be the total weighted fractional cost of B with objective function g that is differentiable and convex (g' is non-decreasing), with $g(0) = 0$. Let $x(t) : \mathbb{R}^+ \to \mathbb{R}^+ \setminus \{0\}$ be any function of t. Let $p_{x,i}^B(t) = (\min(w_i x(t), p_i^B(t)) - q_i^B(t))^+$. Finally, let $Z_x^B(t) = \sum_{i \in Q_B(t)} p_{x,i}^B(t)$. Then,*

$$\int_{t=0}^{\infty} \frac{1}{x(t)} g(Z_x^B(t)/s') \mathrm{d}t \le \frac{1}{s'} B(\sigma).$$

Next we show a property of WSETF that will be useful in relating the volume of work of unsatisfied jobs in WSETF's schedule to that of the optimal solution's schedule. By using this lemma we can bound the volume of jobs in the optimal solution's schedule and then appeal to the lower bound shown in the previous lemma. This lemma is somewhat similar to one shown for the algorithm Shortest-Remaining-Processing-Time (SRPT) [26,19]. However, we are able to get a stronger version of this lemma for WSETF.

Lemma 2. *Consider running WSETF using s-speed for some $s \ge 2$ on m identical machines and the optimal schedule at unit speed on m identical machines. For any job $i \in Q_A(t)$ and time t, it is the case that $Z_i^A(t) - Z_i^O(t) \le 0$.*

Proof. For the sake of contradiction, let t be the earliest time such that $Z_i^A(t) - Z_i^O(t) > 0$. Let j be a job where $p_{i,j}^A(t) > p_{i,j}^O(t)$. Consider the interval $I = [r_j, t]$.

Let I_j be the set of intervals where WSETF works on job j during I and let I'_j be the rest of the interval I. Knowing that $p^A_{i,j}(t) > p^O_{i,j}(t)$, we have that $|I_j| < \frac{1}{s}|I|$. If this fact were not true, then $q^A_j(t) = s|I_j| \geq |I|$, but since OPT has 1 speed, $q^O_j(t) \leq |I|$, and therefore $q^A_j(t) \geq q^O_j(t)$, a contradiction of the definition of job j. Hence, we know that $|I'_j| \geq (1 - \frac{1}{s})|I|$. At each time during I either WSETF is scheduling job j or all m machines in WSETF's schedule are busy scheduling jobs which contribute to $Z^A_i(t)$. Thus the total amount of work done by WSETF during $|I|$ on jobs that contribute to $Z^A_i(t)$ is at least $q^A_j(t) + ms|I'_j| \geq ms(1 - \frac{1}{s})|I| = m(s-1)|I|$. The total amount of work OPT can do on jobs that contribute to $Z^O_i(t)$ is $m|I|$. Let S denote the set of jobs that arrive during I. The facts above imply that

$$\begin{aligned}
Z^A_i(t) - Z^O_i(t) &\leq (Z^A_i(r_j) + \sum_{k\in S} p_{i,k} - m(s-1)|I|) - (Z^O_i(r_j) + \sum_{k\in S} p_{i,k} - m|I|) \\
&= (Z^A_i(r_j) - m(s-1)|I|) - (Z^O_i(r_j) - m|I|) \\
&\leq Z^A_i(r_j) - Z^O_i(r_j) \qquad [s \geq 2] \\
&\leq 0 \quad [t \text{ is the first time } Z^A_i(t) - Z^O_i(t) > 0 \text{ and } r_j < t].
\end{aligned}$$

□

4 Single machine

We now show WSETF is 2-speed $O(1)$-competitive on a single processor for the fractional objective. We then derive Theorem 1. In Section 5, we extend our analysis to bound the performance of WSETF on identical machines as well when migration is allowed.

Assume that WSETF is given a speed $s \geq 2$. Notice that $Z^A_i(t)$ always decreases at a rate of s for all jobs $i \in Q_A(t)$ when $t \in [r_i, C_i]$. This is because $Z^A_i(t)$ is exactly the amount of remaining processing WSETF will do before job i is completed amongst jobs that have arrived by time t. Further, knowing that OPT has 1 speed, we see $Z^O_i(t)$ decreases at a rate of at most 1 at any time t. We know that by Lemma 2 $Z^A_i(r_i) - Z^O_i(r_i) \leq 0$. Using these facts, we derive for any time $t \in [r_i, C^A_i]$,

$$Z^A_i(t) - Z^O_i(t) \leq -(s-1)\cdot(t - r_i).$$

Therefore, $\frac{Z^O_i(t)}{s-1} \geq (t - r_i)$ for any $t \in [r_i, C^A_i]$. Let $a(t)$ denote the job that WSETF works on at time t. By the second definition, WSETF's fractional cost is

$$\int_{t=0}^{\infty} s\cdot\frac{w_{a(t)}}{p_{a(t)}} g(t - r_{a(t)})\mathrm{dt} \leq s\int_{t=0}^{\infty} \frac{w_{a(t)}}{p_{a(t)}} g\Big(\frac{Z^O_{a(t)}(t)}{s-1}\Big)\mathrm{dt} \leq \frac{s}{s-1}\int_{t=0}^{\infty} \frac{w_{a(t)}}{p_{a(t)}} g(Z^O_{a(t)}(t))\mathrm{dt}$$

The last inequality follows since $g(\cdot)$ is convex, $g(0) = 0$, and $\frac{1}{s-1} \leq 1$. By applying Lemma 1 with $x(t) = p_{a(t)}/w_{a(t)}$, $s' = 1$ and B being OPT's schedule, we have the following theorem.

Theorem 8. WSETF *is s-speed $(1+\frac{1}{s-1})$-competitive for the fractional general cost function when $s \geq 2$.*

This theorem combined with Theorem 6 proves Theorem 1.

5 Multiple identical machines

Here we present the proof of Theorem 2. In the analysis of WSETF on a single machine, we bounded the cost of WSETF's schedule for the fractional objective to the cost of the optimal solution for the fractional objective. In the multiple machines case, we will not compare WSETF to the optimal solution for the fractional objective but rather compare to the cost of the optimal solution for the integral objective. We then invoke Theorem 7 to derive Theorem 2. We first consider an obvious lower bound on the optimal solution for the integral objective. For each job i, the best the optimal solution can do is to process job i immediately upon its arrival using one of its m unit speed machines. We know that the total integral cost of the optimal solution is at least

$$\sum_{i\in[n]} w_i g(p_i). \tag{3}$$

Similar to the single machine analysis, when a job is processed we charge the cost to the optimal solution. However, if a job i is processed at time t where $t-r_i \leq p_i$ we charge to the integral lower bound on the optimal solution above. If $t-r_i > p_i$, then we will invoke the lower bound on the optimal solution shown in Lemma 1 and use the fact that the an algorithm's fractional objective is always smaller than its integral objective.

Assume that WSETF is given speed $s \geq 3$. If job $i \in Q_A(t)$ is not processed by WSETF at time t, then there must exist at least m jobs in $Q_A(t)$ processed instead by WSETF at this time. Hence, for all jobs $i \in Q_A(t)$, the quantity $p_i^A(t)+Z_i^A(t)/m$ decreases at a rate of s during $[r_i, C_i^A]$. In contrast, the quantity $Z_i^O(t)/m$ decreases at a rate of at most 1 since OPT has m unit speed machines. Further, by Lemma 2, we know that $Z_i^A(r_i)-Z_i^O(r_i) \leq 0$, and $p_i^A(r_i)+Z_i^A(r_i) - Z_i^O(r_i) \leq p_i$. Using these facts we know for any job i and $t \in [r_i, C_i^A]$ that $p_i^A(t) + (Z_i^A(t) - Z_i^O(t))/m \leq p_i - (s-1)(t-r_i)$. Notice that if $t-r_i \geq p_i$, we have that $p_i^A(t)+(Z_i^A(t)-Z_i^O(t))/m \leq -(s-2)(t-r_i)$. Therefore, $t-r_i \leq \frac{Z_i^O(t)}{m(s-2)}$ when $t-r_i \geq p_i$.

Let $W(t)$ be the set of jobs that WSETF processes at time t. By definition, the value of WSETF's fractional objective is

$$s\int_{t=0}^{\infty} \sum_{i\in W(t)} \frac{w_i}{p_i} g(t-r_i)\mathrm{d}t.$$

We divide the set of jobs in $W(t)$ into two sets. The first is the set of 'young' jobs $W_y(t)$ which are the set of jobs $i \in W(t)$ where $t-r_i \leq p_i$. The other set is

$W_o(t) = W(t) \setminus W_y(t)$ which is the set of 'old' jobs. Let OPT denote the optimal solution's integral cost. We see that WSETF's cost is at most the following.

$$\begin{aligned}
s \int_{t=0}^{\infty} \sum_{i \in W(t)} \frac{w_i}{p_i} g(t - r_i)\mathrm{dt} &\leq s \int_{t=0}^{\infty} \sum_{i \in W_y(t)} \frac{w_i}{p_i} g(t - r_i)\mathrm{dt} + s \int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g(t - r_i)\mathrm{dt} \\
&\leq \int_{t=0}^{\infty} \sum_{i \in W_y(t)} w_i \frac{s}{p_i} g(p_i)\mathrm{dt} + s \int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g(t - r_i)\mathrm{dt} \\
&\leq \sum_{i \in [n]} w_i g(p_i) + s \int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g(t - r_i)\mathrm{dt} \\
&\leq \mathrm{OPT} + s \int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g\Big(\frac{Z_i^O(t)}{m(s-2)}\Big)\mathrm{dt} \\
&\quad \text{[by the lower bound of (3) on OPT]} \\
&\leq \mathrm{OPT} + \frac{s}{s-2} \int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g(Z_i^O(t)/m)\mathrm{dt}
\end{aligned}$$

The third inequality holds since a job i can be in $W_y(t)$ only if i is processed by WSETF at time t, and job i can be processed by at most p_i before it is completed. More precisely, if i is in $W_y(t)$, then it is processed by $s \cdot \mathrm{dt}$ during time $[t, t + \mathrm{dt})$. Hence, $\int_{t=0}^{\infty} \mathbf{1}[i \in W_y(t)] \cdot s \cdot \mathrm{dt} \leq p_i$, where $\mathbf{1}[i \in W_y(t)]$ denotes the 0-1 indicator variable such that $\mathbf{1}[i \in W_y(t)] = 1$ if and only if $i \in W_y(t)$. The last inequality follows since $g(\cdot)$ is convex, $g(0) = 0$, and $\frac{1}{s-2} \leq 1$. We know that a single m-speed machine is always as powerful as m unit speed machines, because a m-speed machine can simulate m unit speed machines. Thus, we can assume OPT has a single m-speed machine. We apply Lemma 1 with $x(t) = p_i/w_i$ for each $i \in W_o(t)$, $s' = m$ and B being OPT's schedule. Knowing that $|W_o(t)| \leq m$, we conclude that $\int_{t=0}^{\infty} \sum_{a \in W_o(t)} \frac{w_a}{p_a} g(Z_a^O(t)/m)$ is at most the optimal solution's fractional cost. Knowing that any algorithm's fractional cost is at most its integral cost, we conclude that WSETF's fractional cost with s-speed is at most $(2 + \frac{2}{s-2})$ times the integral cost of the optimal solution when $s \geq 3$. Using Theorem 7, we derive Theorem 2.

6 Proof of the Main Lemma

In this section we prove Lemma 1.

Proof of [Lemma 1]

The intuition behind the lemma is that each instance of $Z_x^B(t)$ is composed of several infinitesimal job 'slices'. By integrating over how long these slices have left to live, we get an upper bound on $Z_x^B(t)$. We then argue that the integration over each slice's time alive is actually the fractional cost of that slice according to the second definition of the fractional objective. Recall $\beta_i^B(p)$ denotes the latest time t at which $p_i^B(t) = p$. For any time t, let

$$\Lambda_i(t) = \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp},$$

and let $\Lambda(t) = \sum_{i \in Q_B(t)} \Lambda_i(t)$.

The proof of the lemma proceeds as follows. We first show a lower bound on $\Lambda(t)$ in terms of $\frac{1}{x(t)} g(Z_x^B(t)/s')$. Then we show an upper bound on $\Lambda(t)$ in terms of the fractional cost of B's schedule. This strategy allows us to relate $\frac{1}{x(t)} g(Z_x^B(t)/s')$ and B's cost. For the first part of the strategy, we prove that $\frac{s'}{x(t)} g(Z_x^B(t)/s') \le \Lambda(t)$ at all times t. Consider any job $i \in Q_B(t)$ with $p_{x,i}^B(t) > 0$. Suppose $p_i \le w_i x(t)$. Then,

$$\Lambda_i(t) = \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \ge \frac{1}{x(t)} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp}.$$

If $p_i > w_i x(t)$, then by definition of $p_{x,i}^B(t)$,

$$\begin{aligned}
\frac{p_i^B(t)}{p_{x,i}^B(t)} &\ge \frac{p_i^B(t) + q_i^B(t)}{p_{x,i}^B(t) + q_i^B(t)} && [\text{Since } p_i^B(t) \ge p_{x,i}^B(t)] \\
&= \frac{p_i}{\min(w_i x(t), p_i^B(t)) - q_i^B(t))^+ + q_i^B(t)} \\
&\ge \frac{p_i}{(w_i x(t) - q_i^B(t)) + q_i^B(t)} && [\text{Since } p_{x,i}^B(t) > 0] \\
&= \frac{p_i}{w_i x(t)}.
\end{aligned}$$

In this case,

$$\begin{aligned}
\Lambda_i(t) &= \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \\
&\ge \frac{p_i^B(t) w_i}{p_{x,i}^B(t) p_i} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} && [\text{Since } g \text{ is non-decreasing, convex}] \\
&\ge \frac{1}{x(t)} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} && [\text{Since } p_i^B(t)/p_{x,i}^B(t) > p_i/(w_i x(t))].
\end{aligned} \tag{4}$$

In either case, $\Lambda_i(t)$ has a lower bound of quantity (4). By convexity of g, the lower bounds on $\Lambda_i(t)$ are minimized if B completes $p_{x,i}^B(t)$ units of i as quickly as possible for each job i. Schedule B runs at speed s', so we have

$$\Lambda(t) \ge \frac{1}{x(t)} \int_{p=0}^{Z_x^B(t)} g'(p/s')\mathrm{dp} = \frac{s'}{x(t)} \int_{p=0}^{Z_x^B(t)/s'} g'(p)\mathrm{dp} \ge \frac{s'}{x(t)} g(Z_x^B(t)/s').$$

This proves that lower bound on $\Lambda(t)$. Now we show an upper bound on $\Lambda(t)$ in terms of the B's fractional cost. We show $\int_{t=0}^{\infty} \Lambda(t) \mathrm{dt} \leq B(I)$. Fix a job i. We have

$$\int_{t=0}^{\infty} \Lambda_i(t) \mathrm{dt} = \int_{t=0}^{\infty} \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t) \mathrm{dpdt} = \frac{w_i}{p_i} \int_{p=0}^{p_i} \int_{t=0}^{\beta_i^B(p)} g'(t) \mathrm{dtdp}$$
$$= \frac{w_i}{p_i} \int_{p=0}^{p_i} g(\beta_i^B(p)) \mathrm{dp}.$$

By summing over all jobs and using the definition of fractional flow time, we have that $\int_{t=0}^{\infty} \Lambda(t) \mathrm{dt} \leq B(I)$. Further, the given lower bound and upper bounds on $\int_{t=0}^{\infty} \Lambda(t) \mathrm{dt}$ show us that $\int_{t=0}^{\infty} \frac{s'}{x(t)} g(Z_x^B(t)/s') \mathrm{dt} \leq \int_{t=0}^{\infty} \Lambda(t) \mathrm{dt} \leq B(I)$, which proves the lemma. □

7 Lower bounds

We now present the proof of Theorem 3. This lower bound extends a lower bound given in [21]. In [21], it was shown that no oblivious algorithm can be $O(1)$-competitive with speed less than $2-\epsilon$ for the general cost function. However, they assume that the cost function was possibly discontinuous and not convex. We show that their lower bound can be extended to the case where g is convex and continuous. This shows that WSETF is essentially the best oblivious algorithm one can hope for. In all the proofs that follow, we will consider a general cost function g that is continuous, non-decreasing, and convex. The function is also differentiable except at a single point. The function can be easily adapted so that it is differentiable over all points in $\mathbb{R}^+$.

Proof of [Theorem 3]: We appeal to Yao's Min-max Principle [12]. Let A be any deterministic online algorithm. Consider the cost function g and large constant c such that $g(F) = 2c(F - D)$ for $F > D$ and $g(F) = 0$ for $0 \leq F \leq D$. It is easy to see that g is continuous, non-decreasing, and convex. The constant D is hidden to A, and is set to 1 with probability $\frac{1}{2c(n+1)}$ and to $n+1$ with probability $1 - \frac{1}{2c(n+1)}$. Let $\mathcal{E}$ denote the event that $D = 1$. At time 0, one big job J_b of size $n+1$ is released. At each integer time $1 \leq t \leq n$, one unit sized job J_t is released. Here n is assumed to be sufficiently large. That is $n > \frac{12c}{\epsilon^2}$. Note that the event $\mathcal{E}$ has no effect on A's scheduling decision, since A is ignorant of the cost function.

Suppose the online algorithm A finishes the big job J_b by time $n+2$. Further, say the event $\mathcal{E}$ occurs; that is $D = 1$. Since $2n+1$ volume of jobs in total are released and A can process at most $(2-\epsilon)(n+2)$ amount of work during $[0, n+2]$, A has at least $2n+1-(2-\epsilon)(n+2) = \epsilon(n+2)-3$ volume of unit sized jobs unfinished at time $n+2$. A has total cost at least $2c(\epsilon(n+2)-3)^2/2 > c(\epsilon n)^2/2$. The inequality follows since $n > \frac{12c}{\epsilon^2}$. Knowing that $\Pr[\mathcal{E}] = \frac{1}{2c(n+1)}$, A has an expected cost greater than $\Omega(n)$. Now suppose A did not finish J_b by time $n+2$. Conditioned on $\neg\mathcal{E}$, A has cost at least $2c$. Hence A's expected cost is at least $2c(1 - \frac{1}{2c(n+1)}) > c$.

We now consider the adversary's schedule. Conditioned on $\mathcal{E}$ ($D = 1$), the adversary completes each unit sized job within one unit time and hence has a non-zero cost only for J_b. The total cost is $2c(n + 1)$. Conditioned on $\neg\mathcal{E}$ ($D = n + 1$), the adversary schedules jobs in a first in first out fashion thereby having cost 0. Hence the adversary's expected cost is $\frac{1}{2c(n+1)}(2c)(n + 1) = 1$. Knowing that n is sufficiently larger than c, the claim follows since A has cost greater than c in expectation. $\square$

Next we show a lower bound for any non-clairvoyant algorithm that knows g. In [21] it was shown that no algorithm can be $O(1)$-competitive for a general cost function with speed less than $7/6$. However, the cost function g used in the lower bound was neither continuous nor convex. We show that no algorithm can have a bounded competitive ratio if it is given a speed less than $\sqrt{2} > 7/6$ even if the function is continuous and convex but the algorithm is required to be non-clairvoyant.

Proof of [Theorem 4]: Let A be any non-clairvoyant deterministic online algorithm with speed s. Let the cost function g be defined as $g(F) = F - 10$ for $F > 10$ and $g(F) = 0$ otherwise. It is easy to verify that g is continuous, non-decreasing, and convex. At time $t = 0$, job J_1 of processing length 10 units and weight w_1 is released. At time $t = 10(\sqrt{2} - 1)$, job J_2 of weight w_2 is released. Weights of these jobs will be set later. The processing time of job J_2 is set based on the algorithm's decisions, which can be done since the algorithm A is non-clairovyant.

Consider the amount of work done by A on the job J_2 by the time $t = 10$. Suppose algorithm A worked on J_2 for less than $10(\sqrt{2}-1)$ units by time $t = 10$. In this case, the adversary sets J_2's processing time to 10 units. The flow time of job J_2 in A's schedule is $(10-10(\sqrt{2}-1))+(10-10(\sqrt{2}-1))/s \geq 10+10(\sqrt{2}-1)\epsilon/(\sqrt{2}-\epsilon)$ when $s = \sqrt{2}-\epsilon$. Let $\epsilon' = 10(\sqrt{2}-1)\epsilon/(\sqrt{2}-\epsilon)$. Hence, A incurs a weighted flow time of $\epsilon' w_2$ towards J_2. The optimal solution works on J_2 the moment it arrives until its completion, so this job incurs no cost. The optimal solution processes J_1 partially before J_2 arrives and processes it until completion after job J_2 is completed. The largest flow time the optimal solution can have for J_1 is 20, so the optimal cost is upper bounded by $10w_1$. The competitive ratio of A $\frac{\epsilon' w_2}{10 w_1}$ can be made arbitrarily large by setting w_2 to be much larger than w_1.

Now consider the case where A works on J_2 for $10(\sqrt{2} - 1)$ units by time $t = 10$. In this case, the adversary sets the processing time of job J_2 to $10(\sqrt{2}-1)$. Therefore, A completes J_2 by time $t = 10$. However, A can not complete J_1 with flow time of at most 10 units, if given a speed of at most $\sqrt{2}-\epsilon$. Hence A incurs a cost of ϵw_1 towards flow time of J_1. It is easy to verify that for this input, the optimal solution first schedules J_1 until its completion and then processes job J_2 to completion. Hence, the optimal solution completes both the jobs with flow time of at most 10 units, incurring a cost of 0. Again, the competitive ratio is unbounded.

$\square$

Finally, we show a lower bound for any non-clairvoyant algorithm that knows g on m identical machines. We show that no algorithm can have a bounded competitive ratio when given speed less than $2-\frac{1}{m}$. Previously, the only previous lower bounds for the general cost function on identical machines were lower bounds that carried over from the single machine setting.

Proof of [Theorem 5]: We use Yao's min-max principle. Let A be any non-clairvoyant deterministic online algorithm on m parallel machines with the speed $s = 2 - \epsilon$, for any $0 < \epsilon \leq 1$. Let $L > 1$ be a parameter and we take $m > \frac{1}{\epsilon}$. Let the cost function $g(F)$ be defined as follows: $g(F) = F - L$ for $F > L$ and $g(F) = 0$ otherwise. It is easy to verify that, g is continuous, non-decreasing, and convex. At time $t = 0$, $(m-1)L + 1$ jobs are released into the system, out of which $(m-1)L$ jobs have unit processing time and one job has processing time L. The adversary sets the job with processing time L uniformaly at random amongst all the jobs.

Consider the time $t = \frac{L(m-1)+1}{sm}$. At the time t, there exist a job j that has been processed to the extent of at most 1 unit by A since the most work A can do is $smt = L(m-1) + 1$, which is the total number of jobs. With probability $\frac{1}{L(m-1)+1}$, j has a processing time of L units. In the event that j has the processing time of L units, the earliest A can complete j is $t + \frac{L-1}{s} = \frac{L(m-1)+1}{sm} + \frac{L-1}{s} > L$ when L is sufficiently large and $s \leq 2 - \epsilon$ (note that $m > \frac{1}{\epsilon}$). In this case, j has a flow time greater than L time units. Therefore, in expectation A incurs a positive cost.

Let us now look at the adversary's schedule. Since the adversary knows the processing times of jobs, the adversary processes the job j of length L on a dedicated machine. The rest of the unit length jobs are processed on other machines. The adversary completes all the jobs by the time L and hence pays cost of 0. Therefore, the expected competitive ratio of the online algorithm A is unbounded. $\square$

References

1. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: SODA. pp. 1228–1241 (2012)
2. Avrahami, N., Azar, Y.: Minimizing total flow time and total completion time with immediate dispatching. In: SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures. pp. 11–18 (2003)
3. Awerbuch, B., Azar, Y., Leonardi, S., Regev, O.: Minimizing the flow time without migration. SIAM J. Comput. 31(5), 1370–1382 (2002)
4. Azar, Y., Epstein, L., Richter, Y., Woeginger, G.J.: All-norm approximation algorithms. J. Algorithms 52(2), 120–133 (2004)
5. Bansal, N., Chan, H.L.: Weighted flow time does not admit o(1)-competitive algorithms. In: SODA. pp. 1238–1244 (2009)
6. Bansal, N., Krishnaswamy, R., Nagarajan, V.: Better scalable algorithms for broadcast scheduling. In: ICALP (1). pp. 324–335 (2010)
7. Bansal, N., Pruhs, K.: The geometry of scheduling. In: IEE Symposium on the Foundations of Computer Science. pp. 407–414 (2010)

8. Bansal, N., Pruhs, K.: Server scheduling to balance priorities, fairness, and average quality of service. SIAM J. Comput. 39(7), 3311–3335 (2010)
9. Becchetti, L., Leonardi, S.: Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. J. ACM 51(4), 517–539 (2004)
10. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. Journal of Discrete Algorithms 4(3), 339–352 (2006)
11. Bender, M.A., Chakrabarti, S., Muthukrishnan, S.: Flow and stretch metrics for scheduling continuous job streams. In: SODA. pp. 270–279 (1998)
12. Borodin, A., El-Yaniv, R.: On ranomization in online computation. In: IEEE Conference on Computational Complexity. pp. 226–238 (1997)
13. Bussema, C., Torng, E.: Greedy multiprocessor server scheduling. Oper. Res. Lett. 34(4), 451–458 (2006)
14. Chekuri, C., Goel, A., Khanna, S., Kumar, A.: Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In: STOC. pp. 363–372 (2004)
15. Chekuri, C., Im, S., Moseley, B.: Online scheduling to minimize maximum response time and maximum delay factor. Theory of Computing 8(1), 165–195 (2012), `http://www.theoryofcomputing.org/articles/v008a007`
16. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: STOC. pp. 84–93 (2001)
17. Edmonds, J., Im, S., Moseley, B.: Online scalable scheduling for the ℓ_k-norms of flow time without conservation of work. In: ACM-SIAM Symposium on Discrete Algorithms (2011)
18. Edmonds, J., Pruhs, K.: Scalably scheduling processes with arbitrary speedup curves. In: ACM-SIAM Symposium on Discrete Algorithms. pp. 685–692 (2009)
19. Fox, K., Moseley, B.: Online scheduling on identical machines using srpt. In: SODA. pp. 120–128 (2011)
20. Im, S., Moseley, B.: An online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines. In: ACM-SIAM Symposium on Discrete Algorithms (2011)
21. Im, S., Moseley, B., Pruhs, K.: Online scheduling with general cost functions. In: SODA. pp. 1254–1265 (2012)
22. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47(4), 617–643 (2000)
23. Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: A unified approach to scheduling on unrelated parallel machines. J. ACM 56(5) (2009)
24. Leonardi, S., Raz, D.: Approximating total flow time on parallel machines. J. Comput. Syst. Sci. 73(6), 875–891 (2007)
25. Phillips, C.A., Stein, C., Torng, E., Wein, J.: Optimal time-critical scheduling via resource augmentation. Algorithmica 32(2), 163–200 (2002)
26. Pruhs, K., Sgall, J., Torng, E.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chap. Online Scheduling (2004)