Noname manuscript No.
(will be inserted by the editor)

Non-Clairvoyantly Scheduling to Minimize Convex Functions

Kyle Fox · Sungjin Im · Janardhan Kulkarni · Benjamin Moseley

Received: date / Accepted: date

Abstract This paper considers 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. It is 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 $0 < \epsilon < 1$ [22]. This paper shows the first non-trivial results for this problem when g is a non-decreasing convex function 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 $0 < \epsilon < 1$.

K. Fox was 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.
S. Im was supported in part by NSF grant CCF 1409130.
J. Kulkarni was supported by NSF awards CCF-0745761, CCF-1008065, and CCF-1348696.

Kyle Fox
University of Texas at Dallas
E-mail: kyle.fox@utdallas.edu

Sungjin Im
University of California-Merced
E-mail: sim3@ucmerced.edu

Janardhan Kulkarni
University of Minnesota at Twin Cities
E-mail: janardhan.kulkarni@gmail.com

Benjamin Moseley
Carnegie Mellon University
E-mail: moseleyb@andrew.cmu.edu

This paper gives the first non-trivial upper-bound on multiple machines even if the algorithm is allowed to be clairvoyant. All performance guarantees above hold for all non-decreasing convex functions g *simultaneously.* The positive results are supplemented in this paper by lower bounds. The first shows that 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 [27]. In 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 first becomes aware of job i at time r_i when job i is released. This paper assumes jobs can be *preempted*; that is, they can be stopped at any time and later resumed from the previous point of execution. Time is continuous; jobs may have non-integral processing times, and the scheduler may work for a non-integral amount of time on a job before it is preempted or completed.

Generally, a client that submits a job i would like to minimize the *flow time* of the job which is 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 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 this 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 considering these objectives has appealed to the relaxed analysis model called resource augmentation

[23]. 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. An algorithm is said to be *scalable* for an objective if it is $(1+\epsilon)$-speed $O(f(\epsilon))$-competitive for all constant $\epsilon > 0$ where f is some function only dependent of ϵ. For the total weighted flow time objective, it is known that the algorithm Highest-Density-First (HDF) is scalable; that is, it is $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for any fixed $\epsilon > 0$ [26,10]. 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, $\left(\sum_{i\in[n]} F_i^k\right)^{1/k}$ [7,17,20,1,4,24]. 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 jobs. It is known that no algorithm can be $n^{\Omega(1)}$-competitive for minimizing the ℓ_2-norm [7]. 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, $k \geq 1$ [7].

These objectives have also been considered in the identical machine scheduling setting [25,14,3,2,9,16,13,19]. 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 [25]. 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 [22]. In the general cost function problem, a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ is 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. The weight w_i represents job i's priority. 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 [22], 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 $\epsilon > 0$. 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, there is a non-clairvoyant algorithm that optimizes the general cost function objective. Partially addressing this ideal, Im et al. [22] showed Weighted Latest Arrival Processor Sharing (WLAPS) is scalable for concave functions g; however, concave g have the intuitive behavior that each additional unit of time spent waiting on a job is somehow less important than the ones before it. Another open question is that there are currently no nontrivial results known in the multiple identical machines setting or other more general machine environments.

Results: This paper considers 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, this paper gives 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 result, proving that a non-clairvoyant algorithm can simultaneously optimize most reasonable objectives on a single machine (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.*

After establishing this result, this paper considers the general cost function objective on multiple machines for the first time.

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 it is 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. The authors leave this gap as an open problem.

These positive results are complemented by extending the lower bound presented in [22]. 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. This paper shows 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 further 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 $\sqrt{2}$. This establishes that more than $1+\epsilon$ speed is required for an algorithm to be $O(1)$ competitive.

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.*

Comparing to Previous Work: 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 [22] when analyzing HDF. One can think of the fractional objective as converting each job i to a set of p_i unit processing time 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 [22], where the clairvoyant algorithm HDF can assume full knowledge of the conversion. Due to this, in [22] 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 [22] 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.

Technique - A New Lower Bound: The positive results in this paper rely on a new lower bound developed on the optimal solution. This lower bound holds for any objective that is differentiable, non-decreasing, and convex. The 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.

The upper-bounds follow by this lower bound and by bounding the volume of unsatisfied jobs in the optimal schedule at each moment in time. Together, this allows us to relate the optimal solution's objective to WSETF's objective.

The lower bound will likely be useful in further research on scheduling algorithms since it can used for many scheduling objectives. The lower bound is technical and details are given in the main body of the paper.

Other Related Work: For minimizing average flow time on a single machine, the non-clairvoyant algorithms Shortest Elapsed Time First (SETF) and Latest Arrival Processor Sharing (LAPS) are known to be scalable [23, 18]. Their weighted versions Weighted Shortest Elapsed Time First (WSETF) and Weighted Latest Arrival Processor Sharing (WLAPS) are scalable for average weighted flow time [7,6], and also for (weighted) ℓ_k norms of flow time [7,17].

Im et al. [22] 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 [8]. They gave an $O(\log\log P)$-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. It is known that an $O(1)$-approximation exists for the ℓ_k-norms of flow time offline [21]

2 Preliminaries

In this section, we define the algorithm WSETF, define the fractional general cost objective, and introduce some notation used in our analysis. Let g be a positive differentiable, non-decreasing, and convex function where $g(0) = 0$. Let g' denote the derivative of g.

The Algorithm WSETF: First consider the case where there is a single machine. Let $q_i^A(t)$ be the amount WSETF has processed job i by time t. Recall, w_i is the weight of job i. The algorithm WSETF processes the job i such that $\frac{w_i}{q_i^A(t)}$ is maximized. If there are ties, they are broken arbitrarily. In the case that the number of machines $m > 1$, WSETF chooses a set S of at most m jobs that maximizes $\sum_{i\in S} \frac{w_i}{q_i^A(t)}$ and schedules each job on an individual machine. If there are ties, the algorithm breaks them arbitrarily. Throughout our analysis, it is assumed that WSETF works on at most one job at a time on each machine.

The Fractional Objective: We will refer to the non-fractional general cost objective as the *integral* objective. When the schedule is fixed, 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 by charging a job 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:

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

Intuitively, the fractional objective interprets each job as consisting of infinitesimal pieces of equal weight. The first definition measures how much the remaining pieces contribute to the objective at each point in time, while the second measures how much each infinitesimal piece contributes when it is completed.

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

Theorem 6 ([22]) *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 ([22]) *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 from the analysis given in [22]. Theorem 7 in fact follows from Theorem 6 since a schedule's fractional objective is always no greater than its integral objective. We note that the resulting algorithm that performs well for the integral objective is not necessarily the algorithm A. Interestingly, [22] 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. Both of these theorems hold for multiple machines, which could be heterogeneous.

2.1 Notation

We now introduce some more notation that will be used throughout the paper. There are n jobs that arrive over time and each job i has a release time r_i, weight w_i and processing time p_i. 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. We say a job i is *alive or unsatisfied* at time t in schedule B if $r_i \leq t \leq C_i^B$ or $p_i^B(t) > 0$. Let $Q_B(t)$ be the set of jobs released but unsatisfied by B at time t. In words, $Q_B(t)$ is B's queue of jobs at time t.

Let $p_{i,j} = \min\{\frac{w_j}{w_i} p_i, p_j\}$. The value of $p_{i,j}$ is the amount WSETF will processes job j before completing i if i and j are both alive at time $\max\{r_i, r_j\}$. That is, they are both alive at the same time. The value of $p_{i,j}$ can be thought of as the amount of time i will wait while WSETF processes j if i and j arrive at the same time. Let $(\cdot)^+$ denote $\max\{\cdot, 0\}$. Extending the definition of $p_{i,j}$, let $p_{i,j}^B(t) = (p_{i,j} - q_j^B(t))^+$. Notice that $p_{i,j}^B(r_j) = p_{i,j}$ as $q_j^B(r_j) = 0$. If both jobs are alive at time $\max\{r_i, r_j\}$, 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 $[t, C_i^B]$. This is because B will process job j by $p_{i,j}$ amount before completing job i, and it processes $q_j^B(t)$ units of job j by time t. Another interpretation is that $p_{i,j}$ is the remaining time job i waits due to WSETF processing job j.

Let $Z_i^B(t) = \sum_{j \in Q_B(t)} p_{i,j}^B(t)$. In words, the quantity $Z_i^B(t)$ is the remaining amount of time after t that job i will wait to be satisfied assuming that B uses WSETF's scheduling policy and no more jobs arrive. The value of $Z_i^B(t)$ is a lower bound on job i's remaining waiting time, irrespective of later job arrivals, if B uses WSETF's scheduling policy. This is because adding new jobs does not change the relative order of the completion times of existing jobs. Therefore, it cannot change $p_{i,j}^B$ for existing jobs i and j.

When the schedule B is the optimal schedule, we set B to be O and if the schedule is that given by 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. While most of the quantities refer to quantities that correspond to WSETF's scheduling policy, by setting B to be O these quantities allow us to mathematically compare the two schedules.

We let OPT denote the optimal objective. For notational convenience, we also let OPT to denote the optimal scheduler. 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, tools are introduced that are used in the analysis. First, we present our lower bound on the optimal solution. Given a set of partially processed jobs and a time t, the lemma relates the amount of time a fixed job will wait to complete under WSETF's schedule to some amount of the fractional cost incurred in any schedule, including the optimal schedule. This lower bound is a generalization of a lower bound presented in [20] for the norms of flow time. The assumption in the lemma that g is convex is crucial; the lemma is not true otherwise.

Lemma 1 *Let σ be a set of jobs. Let B be any feasible s'-speed schedule of the jobs in σ. Let $B(\sigma)$ 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 $x(t) : \mathbb{R}^+ \to \sigma$ be any function of t to jobs such that $r_{x(t)} \leq t$. Then,*

$$\int_{t=0}^{\infty} \frac{w_{x(t)}}{p_{x(t)}} g(Z^B_{x(t)}(t)/s')\mathrm{d}t \leq \frac{1}{s'} B(\sigma).$$

We first give intuition behind this lemma. As a warm-up, assume that all jobs have unit sizes and weights and set $x(t)$ to the last job in σ that is completed by B. Then, $Z^B_{x(t)}(t)$ is simply the volume of work left in the schedule B at time t assuming that no more jobs arrive; call the quantity Z for notational convenience. (JK: I changed this from OPT schedule to schedule B.) Thus, jobs are completed for the next Z/s time steps. If the schedule B (think of it as the optimal schedule) is non-idling then the left-hand-side is exactly B's fractional cost over s'.

More generally, the lemma measures a certain volume of work that has to be done from time t, that is, $Z^B_{x(t)}$. The fact that this schedule B has $Z^B_{x(t)}$ volume of work implies that some job waits $Z^B_{x(t)}/s'$ time steps to be completed. Further, the schedule will at best complete the fractional pieces of each job contributing to $Z^B_{x(t)}$ during the next $Z^B_{x(t)}/s'$ time steps after time t. The integral is capturing the minimum cost of completing these jobs assuming each such job has density the same as the job $x(t)$. Note this this may not be true, but this is an underestimate for jobs with higher density. For jobs with lower density, the lemma is still true because they actually have much larger size than is contributing to $Z^B_{x(t)}$ and thus are completed even later costing much more since the function g is convex.

Interestingly, the lemma allows us to pick any job $x(t)$ at each time t. Particularly, we can set $x(t)$ to the job our algorithm works on at time t. Hence, even if the optimal scheduler works on a different job at the time, the lemma enables the comparison of our algorithm's cost to the optimal schedule's cost by identifying a volume of work that will incur a comparable cost.

The proof of this lemma is in Section 6. In short, our proof breaks down the jobs contributing to each $Z^B_{x(t)}(t)$ into several infinitesimal job 'slices'. We distribute $Z^B_{x(t)}(t)$ over these slices. Then we integrate over the cost incurred before each slice is completed, and show the result to be an upper bound on $Z^B_{x(t)}(t)$. In turn, we show that the total amount charged to each slice across all $Z^B_{x(t)}(t)$ is at most that slice's contribution to the fractional cost of schedule B.

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. 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 used for the algorithm Shortest-Remaining-Processing-Time (SRPT) [27,19].

Lemma 2 *Consider running WSETF using s-speed for some $s \geq 2$ 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 $Z_i^A(t) - Z_i^O(t) \leq 0$.*

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 $p_{i,j}^A(t) > p_{i,j}^O(t)$; such a job must exist by the assumption that $Z_i^A(t) - Z_i^O(t) > 0$. 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 $p_{i,j}^A(t) > p_{i,j}^O(t)$, we have that $|I_j| < \frac{1}{s}|I|$. To see this, suppose this fact were not true. We have $q_j^A(t) = s|I_j| \geq |I|$. Since the optimal schedule has 1 speed, $q_j^O(t) \leq |I|$, and therefore $q_j^A(t) \geq q_j^O(t)$, a contradiction of the definition of time t and job j. Hence, $|I'_j| \geq (1 - \frac{1}{s})|I|$.

Now we upper bound $Z_i^A(t)$ by taking a close look at how it changes from time r_j. At each time t' during I either WSETF is scheduling job j or all m machines in WSETF's schedule are busy scheduling other jobs which contribute to $Z_i^A(t')$ – since job i is alive at time t', any alive job at the time contributes to $Z_i^A(t')$. These jobs are those contributing to $Z_i^A(r_j)$ (except j), or those arriving during I. Let S denote those latter jobs arriving during I. Note that when a new job k arrives at time t' during I, it adds $p_{i,k}$ to $Z_i^A(t')$. Thus the value of $Z_i^A(t)$ is $Z_i^A(r_j)$ at time r_j and increases by $\sum_{k \in S} p_{i,k}$ as new jobs arrive, and decreases by at least $q_j^A(t) + ms|I'_j| \geq ms(1 - \frac{1}{s})|I| = m(s-1)|I|$. Hence we have $Z_i^A(t) \leq Z_i^A(r_j) + \sum_{k \in S} p_{i,k} - m(s-1)|I|$. Similarly, the value of Z_i^O is $Z_i^O(r_j)$ at time r_j, increases by $\sum_{k \in S} p_{i,k}$, and decreases by at most $m|I|$ since $m|I|$ is the maximum amount of work OPT can do during I. Hence we have $Z_i^O(t) \geq Z_i^O(r_j) + \sum_{k \in S} p_{i,k} - m|I|$.

The facts above imply that

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

□

4 Single machine

We now show WSETF is 2-speed $O(1)$-competitive on a single processor for the fractional objective. Then, Theorem 1 follows from Theorem 6. In Section 5, we extend our analysis to bound the performance of WSETF on identical machines.

Assume that WSETF is given a speed $s \geq 2$. Notice that $Z_i^A(t)$ 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 $Z_i^A(t)$ 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 $Z_i^O(t)$ decreases at a rate of at most 1 at any time t. We know that by Lemma 2, $Z_i^A(r_i) - Z_i^O(r_i) \leq 0$. Using these facts, we derive for any time $t \in [r_i, C_i^A]$,

$$Z_i^A(t) - Z_i^O(t) \leq -(s-1) \cdot (t - r_i).$$

Therefore, $\frac{Z_i^O(t)}{s-1} \geq (t - r_i)$ for any $t \in [r_i, C_i^A]$. Let $x(t)$ denote the job that WSETF works on at time t. By the second definition, WSETF's fractional cost is

$$\begin{aligned}
&\int_{t=0}^{\infty} s \cdot \frac{w_{x(t)}}{p_{x(t)}} g(t - r_{x(t)}) \mathrm{dt} \\
&\leq s \int_{t=0}^{\infty} \frac{w_{x(t)}}{p_{x(t)}} g\Big(\frac{Z_{x(t)}^O(t)}{s-1}\Big) \mathrm{dt} \\
&\leq \frac{s}{s-1} \int_{t=0}^{\infty} \frac{w_{x(t)}}{p_{x(t)}} g(Z_{x(t)}^O(t)) \mathrm{dt}
\end{aligned}$$

The last inequality follows since $g(\cdot)$ is convex, $g(0) = 0$, and $\frac{1}{s-1} \leq 1$. By applying Lemma 1 with $s' = 1$ 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

In this section, the proof of Theorem 2 is presented. In the analysis of WSETF on a single machine, we bounded the cost of WSETF's schedule for the fractional objective by the cost of the optimal solution for the fractional objective. In the multiple machines case, we will compare WSETF to 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. Therefore, 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 and these jobs contribute a positive amount to $Z_i^O(t)$. 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.

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

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

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 at most p_i units before it is completed. More precisely, if i is in $W_y(t)$, then it is processed $s \cdot \mathrm{dt}$ units 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 an m-speed machine can simulate m unit speed machines. Thus, we can assume OPT has a single m-speed machine. Let OPT^f denote the factional cost of the optimal schedule with a m-speed machine. We apply Lemma 1 once for each of the machines that is processing a job in $W_o(t)$. Formally, fix a machine k and let $x_k(t)$ to be the job processed by WSETF on that machine at time t; if the machine is idle at time t, $x_k(t)$ can denote any job that has arrived. Set $x(t) = x_k(t)$, $s' = m$ and B as OPT's schedule in the application of the lemma. Then, Lemma 1 gives that $\int_{t=0}^{\infty} \frac{w_{x_k(t)}}{p_{x_k(t)}} g(Z^O_{x_k(t)}(t)/m)\mathrm{dt} \leq \mathrm{OPT}^f/m$. Therefore, by summing over all machines k, we have $\int_{t=0}^{\infty} \sum_{i \in W_o(t)} \frac{w_i}{p_i} g(Z^O_i(t)/m)\mathrm{dt} \leq \sum_k \int_{t=0}^{\infty} \frac{w_{x_k(t)}}{p_{x_k(t)}} g(Z^O_{x_k(t)}(t)/m)\mathrm{dt} \leq m \cdot \mathrm{OPT}^f/m = \mathrm{OPT}^f$. 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 the proof of Lemma 1 is presented.

Proof Recall $\beta^B_i(p)$ denotes the latest time t at which $p^B_i(t) = p$. For any time $t \geq r_i$, let

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

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 $\frac{w_{x(t)}}{p_{x(t)}} g(Z^B_{x(t)}(t)/s')$ by integrating over the cost of completing slices of jobs contributing to $Z^B_{x(t)}(t)$. 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 $\frac{w_{x(t)}}{p_{x(t)}} g(Z^B_{x(t)}(t)/s')$ and B's cost.

For the first part of the strategy, we prove that $\frac{w_{x(t)}s'}{p_{x(t)}} g(Z^B_{x(t)}(t)/s') \leq \Lambda(t)$ at all times t. Consider any job $j \in Q_B(t)$ with $p^B_{x(t),j}(t) > 0$. Suppose $p_j \leq$

$\frac{w_j}{w_{x(t)}} p_{x(t)}$. Then,

$$\Lambda_j(t) = \frac{w_j}{p_j} \int_{p=0}^{p_j^B(t)} g'(\beta_j^B(p) - t)\mathrm{dp} \geq \frac{w_{x(t)}}{p_{x(t)}} \int_{p=p_j^B(t) - p_{x(t),j}^B(t)}^{p_j^B(t)} g'(\beta_j^B(p) - t)\mathrm{dp}.$$

If $p_j > \frac{w_j}{w_{x(t)}} p_{x(t)}$, then by definition of $p_{x(t),j}^B(t)$,

$$\begin{aligned}
\frac{p_j^B(t)}{p_{x(t),j}^B(t)} &\geq \frac{p_j^B(t) + q_j^B(t)}{p_{x(t),j}^B(t) + q_j^B(t)} \qquad [\text{Since } p_j^B(t) \geq p_{x(t),j}^B(t)] \\
&= \frac{p_j}{(\frac{w_j}{w_{x(t)}} p_{x(t)} - q_j^B(t)) + q_j^B(t)} \\
&= \frac{p_j w_{x(t)}}{w_j p_{x(t)}}.
\end{aligned}$$

In this case,

$$\begin{aligned}
\Lambda_j(t) &= \frac{w_j}{p_j} \int_{p=0}^{p_j^B(t)} g'(\beta_j^B(p) - t)\mathrm{dp} \\
&\geq \frac{p_j^B(t) w_j}{p_{x(t),j}^B(t) p_j} \int_{p=p_j^B(t) - p_{x(t),j}^B(t)}^{p_j^B(t)} g'(\beta_j^B(p) - t)\mathrm{dp} \\
&\qquad [\text{Since } g \text{ is non-decreasing, convex}] \\
&\geq \frac{w_{x(t)}}{p_{x(t)}} \int_{p=p_j^B(t) - p_{x(t),j}^B(t)}^{p_j^B(t)} g'(\beta_j^B(p) - t)\mathrm{dp} \qquad (4) \\
&\qquad [\text{Since } p_j^B(t)/p_{x(t),j}^B(t) \geq p_j w_{x(t)}/(w_j p_{x(t)})].
\end{aligned}$$

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

$$\begin{aligned}
\Lambda(t) &\geq \frac{w_{x(t)}}{p_{x(t)}} \int_{p=0}^{Z_{x(t)}^B(t)} g'(p/s')\mathrm{dp} = \frac{w_{x(t)} s'}{p_{x(t)}} \int_{p=0}^{Z_{x(t)}^B(t)/s'} g'(p)\mathrm{dp} \\
&= \frac{w_{x(t)} s'}{p_{x(t)}} g(Z_{x(t)}^B(t)/s').
\end{aligned}$$

This proves a 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

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

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

7 Lower bounds

We now present the proof of Theorem 3. This lower bound extends a lower bound given in [22]. In [22], it was shown that no cost function 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 a 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. □

Next we show a lower bound for any non-clairvoyant algorithm that knows g. In [22] 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.

□

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 $2 - \frac{1}{m}$. The only previous known lower bounds for the problem on identical machines were the 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 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, the total processing the algorithm A can do is at most smt which is equal to $L(m-1)+1$, the total number of jobs. Note that the algorithm needs to process at least one unit of a job to determine whether the job has processing length L. Since A is deterministic, we can label the jobs 1 through $L(m-1)+1$ independently of the randomness so that randomly choosing some job $k+1$ as the long job means A will complete one unit of jobs 1 through k in that order. With probability $\frac{1}{L(m-1)+1}$, the job with label $L(m-1)+1$ is chosen to be the long job, meaning A does not have time to complete more than one unit of the job. Therefore, with probability $\frac{1}{L(m-1)+1}$, the algorithm does not process the long job more than 1 unit. In this event, the earliest time A can complete the long job 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 $m > \frac{1}{\epsilon}$). Hence, the flow time of the long job is greater than L time units, and 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. S. Anand, N. Garg, and A. Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.
2. N. Avrahami and Y. Azar. Minimizing total flow time and total completion time with immediate dispatching. In *ACM symposium on Parallel Algorithms and Architectures*, pages 11–18, 2003.
3. B. Awerbuch, Y. Azar, S. Leonardi, and O. Regev. Minimizing the flow time without migration. *SIAM J. Comput.*, 31(5):1370–1382, 2002.
4. Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.
5. N. Bansal and H.-L. Chan. Weighted flow time does not admit o(1)-competitive algorithms. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1238–1244, 2009.
6. N. Bansal, R. Krishnaswamy, and V. Nagarajan. Better scalable algorithms for broadcast scheduling. *ACM Trans. Algorithms*, 11(1):3:1–3:24, 2014.
7. N. Bansal and K. Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM J. Comput.*, 39(7):3311–3335, 2010.
8. N. Bansal and K. Pruhs. The geometry of scheduling. *SIAM J. Comput.*, 43(5):1684–1698, 2014.
9. L. Becchetti and S. Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM*, 51(4):517–539, 2004.

10. L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs. Online weighted flow time and deadline scheduling. *J. Discrete Algorithms*, 4(3):339–352, 2006.
11. M. A. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 270–279, 1998.
12. A. Borodin and R. El-Yaniv. On ranomization in online computation. In *IEEE Conference on Computational Complexity*, pages 226–238, 1997.
13. C. Bussema and E. Torng. Greedy multiprocessor server scheduling. *Oper. Res. Lett.*, 34(4):451–458, 2006.
14. C. Chekuri, A. Goel, S. Khanna, and A. Kumar. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In *ACM Symposium on Theory of Computing*, pages 363–372, 2004.
15. C. Chekuri, S. Im, and B. Moseley. Online scheduling to minimize maximum response time and maximum delay factor. *Theory of Computing*, 8(1):165–195, 2012.
16. C. Chekuri, S. Khanna, and A. Zhu. Algorithms for minimizing weighted flow time. In *ACM Symposium on Theory of Computing*, pages 84–93, 2001.
17. J. Edmonds, S. Im, and B. Moseley. Online scalable scheduling for the ℓ_k-norms of flow time without conservation of work. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 109–119, 2011.
18. J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Trans. Algorithms*, 8(3):28:1–28:10, 2012.
19. K. Fox and B. Moseley. Online scheduling on identical machines using SRPT. In *ACM-SIAM Symposium on Discrete Algorithms*, 2011.
20. S. Im and B. Moseley. 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. S. Im and B. Moseley. Fair scheduling via iterative quasi-uniform sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2601–2615, 2017.
22. S. Im, B. Moseley, and K. Pruhs. Online scheduling with general cost functions. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1254–1265, 2012.
23. B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.
24. V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. A unified approach to scheduling on unrelated parallel machines. *Journal of the ACM*, 56(5), 2009.
25. S. Leonardi and D. Raz. Approximating total flow time on parallel machines. *J. Comput. Syst. Sci.*, 73(6):875–891, 2007.
26. C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.
27. K. Pruhs, J. Sgall, and E. Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. 2004.