# Online Non-Preemptive Scheduling to Minimize Maximum Weighted Flow-Time on Related Machines

**Giorgio Lucarelli**
LCOMS, University of Lorraine, Metz, France
giorgio.lucarelli@univ-lorraine.fr

**Benjamin Moseley**
Tepper School of Business, Carnegie Mellon University, USA
moseleyb@andrew.cmu.edu

**Nguyen Kim Thang**
IBISC, Univ. Paris-Saclay, France
kimthang.nguyen@univ-evry.fr

**Abhinav Srivastav**
IBISC, Univ. Paris-Saclay, France
abhinavsriva@gmail.com

**Denis Trystram**
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, France
trystram@imag.fr

**Abstract**

We consider the problem of scheduling jobs to minimize the maximum weighted flow-time on a set of related machines. When jobs can be preempted this problem is well-understood; for example, there exists a constant competitive algorithm using speed augmentation. When jobs must be scheduled non-preemptively, only hardness results are known. In this paper, we present the first online guarantees for the non-preemptive variant. We present the first constant competitive algorithm for minimizing the maximum weighted flow-time on related machines by relaxing the problem and assuming that the online algorithm can reject a small fraction of the total weight of jobs. This is essentially the best result possible given the strong lower bounds on the non-preemptive problem without rejection.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** Online Algorithms, Scheduling, Resource Augmentation

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2019.24

**Funding** *Benjamin Moseley*: Google Research Award and NSF grants CCF-1830711, CCF-1824303, and CCF-1733873
*Nguyen Kim Thang*: OATA ANR-15-CE40-0015-01
*Abhinav Srivastav*: OATA ANR-15-CE40-0015-01

## 1 Introduction

We study the problem of online scheduling non-preemptive jobs to minimize the maximum (or $\ell_\infty$-norm of the) weighted flow-time on related machines. Here, we are given a set of $n$ independent jobs that arrive over time. Each job $j$ has a processing requirement $p_j$ and a weight $w_j$. In the *related machines* environment, each machine $i$ has speed $s_i$ and the time required to process $j$ is equal to $p_j/s_i$. The scheduling algorithm makes online decisions for assigning each job to one of the machines. If a job $j$ arrives at time $r_j$ and completes its processing at time $C_j$, then its flow-time $F_j$ is defined as $(C_j - r_j)$. We focus on the objective


39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019).
Editors: Arkadev Chattopadhyay and Paul Gastin; Article No. 24; pp. 24:1–24:12
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of minimizing the maximum weighted flow-time, *i.e.*, $\max_j w_j F_j$. This metric is often used in systems where jobs are prioritized according to their weights and every job needs to be completed in a reasonable amount of time after its release. The problem of minimizing the maximum flow-time is a natural generalization of the load-balancing problem where jobs arrive over time. This problem is also closely related to deadline scheduling problems.

In this paper, we are interested in designing a non-preemptive schedule whose performance is bounded in the worst-case model. In *non-preemptive* setting, a job, once started processing, must be executed without interruption until its completion time. This is in contrast to *preemptive* setting where a job can be stopped and later continued from where it left off without penalty. Several strong theoretical lower bounds are known for simple instances [9, 4]. In order to overcome this lower bounds, Kalyanasundaram and Pruhs [12] and Phillips et al. [15] proposed the analysis of scheduling algorithms in terms of the speed and machine augmentations, respectively. Together these augmentations are commonly referred to as resource augmentation. In a resource augmentation analysis, the idea is to either give the scheduling algorithm faster processors or extra machines in comparison to the adversary. For preemptive problems, these models have been quite successful in establishing theoretical guarantees on algorithms that achieve good performance in practice [5, 11, 3, 10, 17].

Choudhury et al. [7, 8] proposed a different model of resource augmentation where the online algorithm is allowed to reject a small fraction of the total weight of the incoming jobs, while the adversary must complete all jobs. Theoretically, this model can lead to the discovery of good online algorithms, even in the face of strong lower bounds [7, 13]. Practically, the model is useful for systems where it is assumed that clients loose interest in their job if they wait too long to be completed. Choudhury et al. [7] considered the problem of load balancing as well as the problem of minimizing the maximum weighted flow-time in the restricted assignment setting. In this setting, we are given a set of machines and each job $j$ can only be scheduled on a subset $M_j$ of the machines while its execution takes $p_j$ time units. Even with speed augmentation, these problems admit strong lower bounds in both preemptive and non-preemptive settings. However, online preemptive algorithms that achieve a $\mathcal{O}(1)$-competitive ratio and reject a small fraction of the total weight of jobs have been presented in [7].

Prior works have left open the online *non-preemptive* scheduling problem of minimizing the maximum weighted flow time. Even on a single machine, the problem is not understood and no positive result is known. Moreover, even with speed augmentation, simple examples lead to strong lower bounds. However, recent works on the rejection model [13] give the possibility of creating algorithms with strong guarantees for the non-preemptive setting. Thus, an intriguing open question is whether there exists a constant competitive algorithm for the maximum weighted flow-time objective in the non-preemptive setting assuming that a small fraction of total weight of jobs can be rejected. In this paper, we affirmatively answer this question for the case of related machines by proving the following theorem.

▶ **Theorem 1.** *For the non-preemptive scheduling problem of minimizing the maximum weighted flow-time on related machines, there exists a $\mathcal{O}(1/\epsilon^9)$-competitive algorithm that rejects at most $O(\epsilon)$-fraction of the total weight of jobs, where $\epsilon \in (0,1)$.*

## 1.1 Problem definition and notation

We are given a set $\mathcal{M}$ of $m$ machines and a set $\mathcal{J}$ of $n$ jobs that arrive online. Each machine $i$ processes a job at speed $s_i$. We index the machines such that $s_1 \geq s_2 \geq \ldots s_m$. Each job $j$ is characterized by its release time $r_j$, its processing requirement $p_j$ and its weight $w_j$.

The processing requirement and the weight of $j$ are known at its release time $r_j$. If $j$ is processed on machine $i$, then it requires $p_j/s_i$ time units. The goal is to schedule the jobs *non-preemptively*. Given a schedule $\mathcal{S}$, the completion time of a job $j$ is denoted by $C_j^{\mathcal{S}}$. The weighted flow-time of $j$ is defined as $w_j F_j^{\mathcal{S}} = w_j(C_j^{\mathcal{S}} - r_j)$, which is the weighted amount of time during which $j$ remains in the system. The objective is to minimize the maximum ($\ell_\infty$-norm of) weighted flow-time, *i.e.*, $\max_j w_j F_j^{\mathcal{S}}$. We omit the superscripts if the schedule $\mathcal{S}$ is clearly defined by the context.

Let $F$ denote the value of the offline optimal solution. Let $\epsilon$ be an arbitrary constant such that $\epsilon \in (0,1)$. We assume that all weights $w_j$ are of the form $(1/\epsilon)^k$, where $k$ is an integer. This can be assumed by *rounding down* weights to the nearest power of $\frac{1}{\epsilon}$ which affects the competitive ratio by a factor of at most $\left(\frac{1}{\epsilon}\right)$. After rounding, we say that a job $j$ is of *class* $k$ if $w_j = \left(\frac{1}{\epsilon}\right)^k$. Let $K$ denote the largest weight class of any job. A job $j$ is *valid* on machine $i$, iff it takes at most $F/w_j$ time units on $i$, that is $\frac{p_j}{s_i} \leq \frac{F}{w_j}$.

## 1.2 Organization

In Section 2, we present the works related to our problem. Then, in Section 3, we present an offline algorithm for the scheduling problem of minimizing the maximum weighted flow-time on related machines. This algorithm is inspired by Anand et al. [2] and uses a small look-ahead, allows preemptions and does not respect the release dates. Specifically, we assume that the value $F$ of the optimal weighted flow time is known to the algorithm. Since release dates are not respected, the algorithm creates an infeasible schedule. Later, in Section 4, we discuss how to convert this offline algorithm to an online algorithm respecting the non-preemptive requirement. Note that the release dates will be respected due to the online nature. Finally, we explain how to remove the assumption about knowing the value $F$ in Section 5.

# 2 Related Work

We discuss first related works for the unweighted case. For a single machine, First-In-First-Out is an optimal algorithm for minimizing the maximum flow-time. For identical machines, Bender et al. [14] and Ambulh and Mastrolilli [1] showed that the algorithm that schedules the incoming jobs on the least loaded machine, is $(3-2/m)$-competitive. On related machines, Bansal et al. [4] showed that there exists a 13.5-competitive algorithm. This has recently been improved to a 12.5-competitive algorithm by Im et al [16]. All the above algorithms are non-preemptive and their results hold against both the preemptive and non-preemptive adversary. For the more general setting of unrelated machines, Anand et al. [2] gave an $\mathcal{O}(1/\epsilon)$-competitive algorithm with $(1+\epsilon)$-speed augmentation and this result fundamentally uses preemption.

In the presence of weights, only results in the preemptive setting are known for the problem of minimizing the maximum weighted flow-time. Bender et al. [14] showed a lower bound of $\Omega(P^{1/3})$ on the competitive ratio on single machine where $P$ is the ratio of the minimum to maximum job size. This was later improved to $\Omega(P^{0.4})$ in [6]. Both these lower bounds also hold if $P$ is replaced with the ratio of the maximum to minimum weight. In speed augmentation model, Bansal and Pruhs [5] showed that the Highest Density First policy is $(1+\epsilon)$-speed $\mathcal{O}(1/\epsilon^2)$-competitive on a single machine. Chekuri and Moseley [6] presented a $(1+\epsilon)$-speed $\mathcal{O}(1/\epsilon)$-competitive algorithm for parallel machines, while Anand et al. [2] proposed a $(1+\epsilon)$-speed $\mathcal{O}(1/\epsilon^3)$-competitive algorithm for related machines. In the rejection model, Choudhury et al. [7] presented an $\mathcal{O}(1/\epsilon^4)$-competitive algorithm for the restricted assignment settings when an $\epsilon$-factor of the weight of the jobs can be rejected by the online algorithm.

## 3 An Offline Look-ahead Algorithm with Preemptions

In this section we assume that the value $F$ of the optimal solution is given, preemptions are allowed and the release dates of the jobs are not necessarily respected. Intuitively, we show the following. For ease of explanation assume that all jobs have unit weight. We consider all the jobs released during a long interval of size $O(F/\epsilon)$. Since the maximum weighted flow-time is $F$, all such jobs must be scheduled within an interval of size $O(F/\epsilon + F)$ by the optimal solution. We show that by rejecting an $O(\epsilon)$-fraction of the total weight of jobs, an online algorithm can schedule all the remaining jobs in the interval of size $O(F/\epsilon)$. The algorithm below builds on this idea when jobs have different weights. This is inspired by the work of Anand et al. [2] where speed augmentation is used to achieve a similar effect. Recall that there exists a strong lower bound in the speed augmentation model. In rejection model, one needs to ensure that the algorithm rejects at most $O(\epsilon)$-fraction of jobs both in terms of weights and volume.

We allow our algorithm to reject some jobs. For each weight class $k$ and integer $\ell$, let $I(k,\ell)$ denote the interval $\left[\frac{\ell F \epsilon^k}{\epsilon^3}, \frac{(\ell+1)F\epsilon^k}{\epsilon^3}\right)$. We say that a job $j$ belongs to type $(k,\ell)$ if it is of class $k$ and $r_j \in I(k,\ell)$. Observe that intervals $I(k,\ell)$ form a nested set of intervals. Note that at least $\left(\frac{1}{\epsilon^3}\right)$ jobs that belong to class $k$ or more, can be scheduled during an interval $I(k,\ell)$. The online algorithm $\mathcal{A}$ is defined to have the following rejection and scheduling policies.

**Rejection policy.** The rejection policy of $\mathcal{A}$ is described in Algorithm 1. The algorithm uses a simple rejection policy where it ensures that for each interval $I(k,\ell)$ the algorithm rejects at least $\epsilon^2/2$-fraction of volume of jobs and $\mathcal{O}(\epsilon)$-fraction of weight of jobs.

**Algorithm 1** $R_{\mathcal{A}}(\mathcal{I}, F, \epsilon)$.

**for** $k = K$ to 1 **do**
  **for** $\ell = 1, 2, \ldots$ **do**
    $J(k,\ell) :=$ the set of jobs of type $(k,\ell)$
    $D := \lfloor \epsilon^2 |J(k,\ell)| + \epsilon \sum_{\substack{I(k'\ell') \subseteq I(k,\ell): \\ k'=k+1}} |J(k',\ell')| \rfloor + \sum_{\substack{I(k',\ell') \subseteq I(k,\ell): \\ k' \geq k+2}} |J(k',\ell')|$
    Reject longest-$D$ jobs from $J(k,\ell)$

**Scheduling policy.** The scheduling policy of $\mathcal{A}$ is described in Algorithm 2. The algorithm uses the following order: it picks jobs in the decreasing order of their class, and within each class it goes by increasing order of its intervals. When considering a job $j$, the algorithm schedules $j$ during the interval $I(k,\ell)$ on the slowest *valid* machine with enough free space. Jobs may be scheduled preemptively. This completes the description of the algorithm $\mathcal{A}$.

**Algorithm 2** $S_{\mathcal{A}}(\mathcal{I}, F, \epsilon)$.

**for** $k = K$ to 1 **do**
  **for** $\ell = 1, 2, \ldots$ **do**
    **for** each non-rejected job $j$ of type $(k,\ell)$ **do**
      $m_j :=$ the slowest machine for which $j$ is valid
      **for** $i := m_j, \ldots, 1$ **do**
        If there is at least $p_j/s_i$ free slots (preemptive) on machine $i$ during $I(k,\ell)$ then schedule $j$ on $i$ during the first such free slots.

## 3.1 Analysis of the Offline Algorithm

In this section, we prove that Algorithm $S_{\mathcal{A}}$ will always find enough space to schedule the non-rejected set of jobs in $R_{\mathcal{A}}$.

▶ **Theorem 2.** *Algorithm $S_{\mathcal{A}}$ outputs a preemptive schedule for the set of non-rejected jobs which ensures that each job that belongs to type $(k, \ell)$ is scheduled during $I(k, \ell)$. Note that schedule may process jobs before their release date.*

We prove this by contradiction. Let $j^*$ be the first non-rejected job that algorithm $\mathcal{A}$ cannot schedule on some machine $i$. Then we will show that the value of the offline optimal solution is strictly greater than $F$, which contradicts our assumption on the knowledge of the value of optimal offline solution, $F$.

Assume that $j^*$ is of type $(k^*, \ell^*)$. We build a set $S$ of job recursively. Initially $S$ just contains $j^*$. We add $j'$ of type $(k', \ell')$ to $S$ if there is already a job $j$ of type $(k, l)$ in $S$ satisfying the following conditions:

1. $k' \geq k$.
2. $\mathcal{A}$ processes $j'$ on a machine $i$ which is valid for $j$ as well.
3. $\mathcal{A}$ processes $j'$ during the $I(k', \ell')$ such that $I(k', \ell') \subseteq I(k, \ell)$

For a machine $i$ and interval $I(k, \ell)$, define the *machine-interval* $I_i(k, \ell)$ as the time interval $I(k, \ell)$ on machine $i$. We construct a set $\mathcal{I}_M$ of machine-intervals as follows: For every job $j \in S$ of type $(k, \ell)$, we add the interval $I_i(k, \ell)$ to $\mathcal{I}_M$ for all machines $i$ such that $j$ is valid for $i$.

▶ **Definition 3.** *We say that an interval $I_i(k, \ell) \in \mathcal{I}_M$ is* maximal *if there is no other interval $I_i(k', \ell')$ which contains $I_i(k, \ell)$.*

Observe that every job in $S$ except $j^*$ gets processed in one of machine-intervals in $\mathcal{I}_M$. Let $\mathcal{I}_X$ denote the set of maximal intervals in $\mathcal{I}_M$. We show that the jobs in $S$ satisfy the following property.

▶ **Lemma 4.** *For any maximal interval $I_i(k, \ell) \in \mathcal{I}_X$, Algorithm $S_{\mathcal{A}}$ processes a job on at least $(1 - \epsilon^3)$-fraction of the interval on machine $i$.*

**Proof.** We prove this property holds whenever we add a new maximal interval to $\mathcal{I}_X$. Suppose this property holds at some point in time, and we add a new job $j'$ to $S$. Let $j, k, \ell, j', k', \ell'$ be as in the description of $S$. Since $k' \geq k$ and $j$ is valid for $i$, the interval set $\mathcal{I}_X$ already contains the interval $I_{i'}(k, \ell)$ for all $i' \leq i$. Hence the intervals $I_{i'}(k', \ell')$ cannot be maximal for any $i' \leq i$. Suppose an interval $I_{i'}(k', \ell')$ is maximal, where $i' > i$, and $j'$ is valid for $i'$. Our algorithm would have considered scheduling $j'$ on $i'$ before going to $i$. Hence the machine $i'$ is at least $|I_{i'}(k', \ell')| - p_j/s_{i'}$ amount busy scheduling other jobs from $S$. The lemma follows since $p_j/s_{i'} \leq F/w_j \leq F\epsilon^k$. ◀

▶ **Corollary 5.** *There are at least $(\frac{1}{\epsilon^3} - 1)$ jobs of class $k$ or more scheduled for every $I_i(k, \ell) \in \mathcal{I}_X$.*

**Proof.** Recall that the size of the interval $I_i(k, \ell)$ is $\frac{F\epsilon^k}{\epsilon^3}$ and the size of the longest job scheduled in the interval $I_i(k, \ell)$ is $\epsilon^k F$. Combining these facts with Lemma 4 shows that the corollary holds. ◀

Next we associate the set of rejected jobs to the maximal intervals. Recall that $|I(k, \ell)|$ denote the length of the interval $I(k, \ell)$. Intuitively, we show that for each maximal interval $I_i(k, \ell) \in \mathcal{I}_X$, we can associate at least $\mathcal{O}(\epsilon^2)|I_i(k, \ell)|$ volume of jobs that are rejected by the algorithm $R_{\mathcal{A}}$ such that these jobs are of type $(k', \ell')$ where $I(k', \ell') \subseteq I(k, \ell)$. To this end, let $R$ denote the set of job rejected by $R_{\mathcal{A}}$. Let $R(k, \ell) = \{j \in R : j \text{ is of type } (k', \ell') \text{ and } I(k', \ell') \subseteq I(k, \ell)\}$.

▶ **Lemma 6.** *There exists a function $\phi : R \to \mathcal{I}_X$ such that for every $I_i(k, \ell) \in \mathcal{I}_X$, it holds that $vol(\phi^{-1}(I_i(k, \ell))) \geq \frac{\epsilon^2}{4}|I_i(k, \ell)|$ and $\phi^{-1}(I_i(k, \ell)) \subseteq R(k, \ell)$ where $vol(Q)$ denotes the total volume of jobs in the set $Q$.*

**Proof.** Fix a maximum interval $I = I_i(k, \ell)$. Let $k_{max}$ denote the maximum weight class of the job scheduled in $I$. Recall the intervals $I_i(k', \ell') \subseteq I_i(k, \ell)$ are nested.

We first form an $1/\epsilon$-ary tree where a node $v(k', \ell')$ represents the set of jobs of type $(k', \ell')$ scheduled in the interval $I$ on $i$. The node $v(k', \ell')$ is the the ancestor of the node $v(k'+1, \ell'')$ iff $I_i(k'+1, \ell'') \subseteq I_i(k', \ell')$. The height of this tree is $k_{max} - k$. Note that some of the leaves can be empty. Therefore, we trim the tree such that leaves are non-empty. For this, we find an empty leave and remove it from the tree. We repeat this procedure until no empty leaves are present. Note that an intermediate node of the tree can be empty. Next, we consider the following cases depending upon the number of jobs in the leaves:

**Case 1:** *There are at least $1/\epsilon^2$-jobs in each leaf.* The algorithm $R_{\mathcal{A}}$ rejects at least $\epsilon^2/2$ number of jobs at each non-empty node of the tree. Let $j$ be such a job rejected by $R_{\mathcal{A}}$ for some node in the tree, then we define $\phi(j) = I$ (i.e., associate rejected job $j$ to interval $I$). Recall that $R_{\mathcal{A}}$ rejects longest jobs among jobs of fixed class. Thus, the total volume of jobs associated with the interval $I$ is at least $\epsilon^2/2$ and the lemma holds.

**Case 2:** *If the number of jobs in each leaf is between $1/\epsilon$ and $1/\epsilon^2$.* The algorithm $R_{\mathcal{A}}$ rejects at least $\epsilon^2/2$ fraction of volume of jobs at each non-empty node except leaves. As before, let $j$ be such a job rejected by $R_{\mathcal{A}}$, then we define $\phi(j) = I$. We show that the total volume of jobs in the leaves are small. Let $v(k', \ell')$ denote jobs corresponding to some leaf. Then $|v(k', \ell')| < 1/\epsilon^2$. The total volume of jobs in $v(k', \ell')$ is at most $(F\epsilon^{k'})/\epsilon^2 = \epsilon|I_i(k', \ell')|$. Observe that the jobs of any two leaves are scheduled independent of each other in separate sub-intervals. Combining this fact with the previous bound on the volume of jobs in leaves implies that the total volume of jobs in leaves is at most $\epsilon|I|$. Thus, the total volume of jobs scheduled during the interval $I$ is at most $2.vol(\phi^{-1}(I))/\epsilon^2 + \epsilon|I|$. Since $S_{\mathcal{A}}$ processes jobs on at least $(1-\epsilon^3)$-fraction of $I$, it holds that $vol(\phi^{-1}(I)) \geq (\epsilon^2/2)(1-\epsilon^3-\epsilon)|I| \geq (\epsilon^2/4)|I|$.

**Case 3:** *If the number of jobs in each leaf is strictly less than $1/\epsilon$.* If the algorithm rejects $\epsilon^2$-fraction of total volume of jobs at each non-empty level other than the leaf, then the lemma holds (the proof is similar to Case 2). Thus, we consider the case where the parent of a leaf does not reject $\epsilon^2$-fraction of the total volume of jobs. This implies that each parent has at most $1/\epsilon^2$ number of jobs and the height of the subtree rooted at the parent node is at most 1. The algorithm $R_{\mathcal{A}}$ rejects $\epsilon^2/2$ jobs for each node from the root to the parent of parent of a leaf. As before, let $j$ be such a job rejected by $R_{\mathcal{A}}$, then we define $\phi(j) = I$. Unlike Case 2 where intervals corresponding to leaves are disjoint, th intervals of parents of two leaves can overlap. Here, we use the top-down approach to count the total volume of jobs. Each job in the parent node is split into $1/\epsilon$-parts. We "virtually force" these parts to be accounted in the leaves of that parent (even though their weight class is strictly smaller than the weight class of the leaves). Thus the number of jobs in each leaf can increase by at most $1/\epsilon^2$. Using arguments similar to Case 2, the total volume of jobs in the leaves is at most $2\epsilon I$. Since $S_{\mathcal{A}}$ process job on at least $(1-\epsilon^3)$-fraction of $I$, it holds that $vol(\phi^{-1}(I))$ is at least $\epsilon^2/2(1-\epsilon^3-2\epsilon)|I|$-volume of jobs. ◀

▶ **Corollary 7.** *The total volume of jobs in* $S' = S \cup R$ *is greater than* $\sum_{I(k,\ell)\in\mathcal{I}_{\mathcal{X}}} I(1+\epsilon^3)|I(k,\ell)|$.

▶ **Lemma 8.** *If the value of offline solution is at most* $F$*, then the total volume of jobs in* $S'$ *is at most* $\sum_{I(k,\ell)\in\mathcal{I}_{\mathcal{X}}}(1+\epsilon^3)|I(k,\ell)|$.

**Proof.** For any maximal interval $I(k,\ell)$ on machine $i$, let $I_i^{\epsilon}(k,\ell)$ be the interval of length $(1+\epsilon^3)|I(k,\ell)|$ which starts at the same time as $I(k,\ell)$ on machine $i$.

Let $j \in S$ be a job of type $(k,\ell)$. The optimal offline solution must schedule $j$ within $F\epsilon^k$ of its release date. Since $r_j \in I(k',\ell') \subseteq I(k,\ell)$, the optimal solution must process a job $j$ during $I^{\epsilon}(k,\ell)$. So, the total volume of jobs in $S$ can be at most $|\bigcup_{I(k,\ell)\in\mathcal{I}_{\mathcal{X}}} I_i^{\epsilon}(k,\ell)| \leq \sum_{I(k,\ell)\in\mathcal{I}_{\mathcal{X}}}(1+\epsilon^3)|I(k,\ell)|$. ◀

Clearly, Corollary 7 contradicts Lemma 8. So, Algorithm $S_{\mathcal{A}}$ must be able to process all the jobs.

▶ **Lemma 9.** *The total weight of jobs rejected by the algorithm* $R_{\mathcal{A}}$ *is* $\mathcal{O}(\epsilon)$*-fraction of the total weight of jobs in the instance* $\mathcal{I}$.

## 4 The Online Algorithm $\mathcal{B}$

The previous algorithm $\mathcal{A}$ is an offline preemptive algorithm for $\mathcal{I}$ that does not respect the release dates. This section presents an online non-preemptive algorithm $\mathcal{B}$. This algorithm is assumed to know the optimal objective $F$ and this algorithm is extended in a later section to when this is not known. The algorithm maintains a queue for each machine $i$ and time $t$. Unlike the previous algorithm, $\mathcal{B}$ rejects the job of type $(k,\ell)$ at the end of the interval $I(k,\ell)$. For each non-rejected job $j$, $\mathcal{B}$ uses $S_{\mathcal{A}}$ to figure out the assignment of jobs to the machines. This algorithm differs from the online algorithm mentioned in Anand et al. [2] as it schedules jobs non-preemptively and does not necessarily process jobs in their decreasing order of their weights.

The rejection and assignment policies of $\mathcal{B}$ in given Algorithm 3.

**Algorithm 3** $M_{\mathcal{B}}(\mathcal{I}, F, \epsilon)$.

**for** $t = 0, 1, 2, \cdots$ **do**
  Let $K$ denote the largest class of a job.
  **for** $k = K$ to 1 **do**
    **if** $t$ is the end point of the interval $I(k,\ell)$ for some $\ell$ **then**
      **Rejection similar to $R_{\mathcal{A}}$**
      $J(k,\ell) :=$ the set of jobs of type $(k,\ell)$
      $D := \lfloor \epsilon^2|J(k,\ell)| + \epsilon \sum_{\substack{I(k'\ell')\subseteq I(k,\ell):\\ k'=k+1}} |J(k',\ell')| \rfloor + \sum_{\substack{I(k',\ell')\subseteq I(k,\ell):\\ k'\geq k+2}} |J(k',\ell')|$
      Reject longest-$D$ Tjobs from $J(k,\ell)$ from the remaining jobs in $J(k,\ell)$
    **Assignment similar to $S_{\mathcal{A}}$**
    **for** each non-rejected job $j$ of class $k$ **do**
      Let $m_j$ denote the machine on which $j$ is scheduled by $S_{\mathcal{A}}$
      Assign $j$ to the queue of $m_j$

After the execution of Algorithm 3, the algorithm $\mathcal{B}$ uses two more rejection policies for each machine $i$ . The first policy ensures that $\mathcal{B}$ rejects $\mathcal{O}(\epsilon^2)$-fraction of new assigned jobs whereas the second policy ensures that $\mathcal{B}$ processes jobs in a non-preemptive fashion. At any time if the machine $i$ is idle, $\mathcal{B}$ picks a job from the highest class according to the ordering given by $S_{\mathcal{A}}$.

**Making $\mathcal{B}$ Non-preemptive.** We now detail the second rejection policy. During the processing of a job of some class on a machine $i$, the algorithm maintains a bound on total weight of higher class jobs that are newly assigned to machine $i$. Let $j$ be the job running at the start of interval $I(k, \ell+1)$ on machine $i$. Let $k_j$ denote the class of $j$. $\mathcal{B}$ rejects $j$ if there is a new job $j'$ of type $(k', l')$ that $k' \geq k_j + 2$ and the intervals $I(k', \ell')$ and $I(k, \ell)$ end at same time. This ensures that the weight of job $j$ and $j'$ differ at least by a factor of $1/\epsilon$. It may happen that there is no job class $k' \geq k_j + 2$. In this case, the algorithm $\mathcal{B}$ rejects $j$ if there are at least $(1/\epsilon$ newly arrived jobs of type $(k', \ell')$ if $k' \geq k_j + 1$ and the intervals $I(k', \ell')$ and $I(k, \ell)$ end at same time. Note that this also ensures the weight of newly arrived jobs is at least an $(1/\epsilon)$ times the weight of current running job. These rejection policies and scheduling policy of the algorithm $\mathcal{B}$ for the machine $i$ at time $t$ is mentioned in Algorithm 4.

**Algorithm 4** $S_{\mathcal{B}}(\mathcal{I}, F, \epsilon, i, t)$.

**Rejection similar to $R_{\mathcal{A}}$**
**for** $k = K$ to 1 **do**
  **if** $t$ is the end point of the interval $I(k, \ell)$ for some $\ell$ **then**
    $J_i(k, \ell) :=$ the set of jobs of type $(k, \ell)$ assigned to $i$ at $t$
    $D := \lfloor 2\epsilon^2 |J_i(k,\ell)| + 2\epsilon \sum_{\substack{I(k'\ell') \subseteq I(k,\ell): \\ k'=k+1}} |J_i(k', \ell')| \rfloor + 2 \sum_{\substack{I(k',\ell') \subseteq I(k,\ell): \\ k' \geq k+2}} |J_i(k', \ell')|$
    Reject $D$-longest jobs from $J(k, \ell, i)$

**Making algorithm non-preemptive**
Let $j \in (k, \ell)$ be the job executing on $i$ at $t$
Let $J_{k'}$ denote the set of jobs of class $k'$ assigned to $i$ at $t$
**if** $|J_{(k+1)}| \geq 1/\epsilon$ or $\exists k'' : |J_{(k'')}| > 0$ and $k'' \geq k+2$ **then**
  Reject $j$

**Scheduling Policy**
**if** the machine $i$ is idle **then**
  Start processing the earliest job of highest class in the queue of $i$

## 4.1 Analysis

For a class $k$, let $J_k$ be the jobs of class at least $k$. For a class $k$, integer $\ell$, and machine $i$, let $Q(i, k, \ell)$ denote the jobs of $J_k$ which are in the queue of machine $i$ at the beginning of $I(k, \ell)$. The jobs in $Q(i, k, \ell)$ could consist of either

1. jobs in $Q(i, k, \ell - 1)$, or
2. jobs of $J_k$ which get processed by $\mathcal{A}$ during $I(k, \ell - 1)$ on machine $i$. Indeed, the jobs of $J_k$ which are dispatched to machine $i$ during $I(k, \ell - 1)$ will complete processing in $I(k, \ell - 1)$ in $\mathcal{A}$ and hence may get added (if not rejected) to $Q(i, k, \ell)$. Let $P(i, k, \ell - 1)$ denotes the volume of such jobs that are added by $\mathcal{B}$ to the queue of machine $i$.

Next, we note some properties of the algorithm $\mathcal{B}$:

▶ **Property 1.** *A job $j$ gets scheduled in $\mathcal{B}$ only in later slots than those it was scheduled on by $\mathcal{A}$.*

▶ **Property 2.** *For a class $k$, integer $\ell$ and machine $i$, the total processing of jobs in $P(i,k,\ell)$ is at most $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$.*

**Proof.** If the volume of jobs processed by algorithm $\mathcal{A}$ during the interval $I(k,\ell)$ is at most $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$, then the property holds trivially. Assume that the volume of jobs processed by algorithm $\mathcal{A}$ during the interval $I(k,\ell)$ is strictly greater than $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$. Then it holds that the algorithm rejects at least $\epsilon^2/4$-fraction of volume of jobs assigned to $i$ (the proof is similar to Lemma 6). Thus the total volume of jobs assigned to $i$ is strictly greater than $\frac{(1+\epsilon^3)F\epsilon^k}{\epsilon^3}$. But, this contradicts Theorem 2. ◀

▶ **Property 3.** *For a class $k$, integer $l$ and machine $i$, the total remaining processing time of jobs in $Q(i,k,\ell)$ is at most $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$.*

**Proof.** We use induction. Suppose this is true for some $i,k,l$. We show that this holds for $i,k,\ell+1$ as well. By induction $|Q(i,k,\ell)|$ is at most $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$. We consider multiple separate cases based on which job gets processed during the interval $I(k,\ell)$ on machine $i$.

1. *Suppose the machine $i$ is busy processing jobs from $J_k$ during $I(k,\ell)$.*
   Then algorithm either processes job from $Q(i,k,\ell)$ or $P(i,k,\ell)$. The total volume of such jobs are bounded by $\frac{2(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$. The property holds since the total volume of job processed by $i$ is $|I(k,\ell)| = \frac{F\epsilon^k}{\epsilon^3}$.
2. *Suppose job $j$ of class smaller than $k$ is processed at the start of $I(k,\ell)$ and $Q(i,k,\ell)=0$.*
   In this case, $Q(i,k,\ell+1)$ consists of the jobs of $P(i,k,\ell)$. The property follows since $|P(i,k,\ell)| \le \frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3}$.
3. *Suppose job $j$ of class smaller than $k$ is processing at the start of $I(k,\ell)$ and $Q(i,k,\ell)>0$.*
   We show that $Q(i,k,\ell)$ is at most $\frac{F\epsilon^k}{\epsilon}$. Let $\sigma_j$ denote the starting time of job $j$ on machine $i$. Then we have that $\sigma_j > \frac{\ell F\epsilon^k}{\epsilon^3} - p_j/s_i \ge \frac{\ell F\epsilon^k}{\epsilon^3} - \frac{F\epsilon^k}{\epsilon}$. Since algorithm $\mathcal{B}$ prefers jobs of higher class, it must be the case that at $\sigma_j$ no job of class $k$ or higher was available with machine $i$ . Hence $Q(i,k,\ell)$ consists of jobs that were added to the queue of machine $i$ during the interval $\left(\sigma_j, \frac{\ell F\epsilon^k}{\epsilon^3}\right]$. Since $Q(i,k,\ell)>0$ and the class of $j$ is strictly smaller than $k$, $j$ must belong of class $(k-1)$, otherwise $\mathcal{B}$ would reject $j$ due to non-preemptive rejection policy. Moreover, there are at most $1/\epsilon$ jobs of class $k$ in $Q(i,k,\ell)$. Hence, the total volume of jobs in $Q(i,k,\ell)$ is most $\frac{F\epsilon^k}{\epsilon}$. The property follows from the facts that $\mathcal{B}$ spends at most $\frac{F\epsilon^k}{\epsilon}$ processing time on $j$.
4. Suppose $\mathcal{B}$ processes a job of class smaller than $k$ at some point in $I(k,\ell)$.
   This implies that $Q(i,k,\ell+1)$ contains jobs that are released during the interval $I(k,\ell)$. The property holds due to Claim 2. ◀

▶ **Theorem 10.** *In the schedule $\mathcal{B}$ a job $j$ of class $k$ has flow-time at most $\frac{F\epsilon^k}{\epsilon^8}$. Hence the algorithm $\mathcal{B}$ is an $O(\frac{1}{\epsilon^9})$-competitive algorithm that rejects at most $O(\epsilon)$-fraction of total weights of job.*

**Proof.** Consider a job $j$ of class $(k,\ell-1)$. Suppose it gets processed on machine $i$. The algorithm $\mathcal{B}$ adds $j$ to the queue $Q(i,k,\ell)$. Let $j'$ be the job running at beginning of the interval $I(k,\ell)$. Property 3 from above implies that the total remaining processing time of jobs in $Q(i,k,\ell)$ is at most $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^3} = (1-\epsilon^3)|(k,\ell)|$.

Consider an interval $I$ that starts at same time as $I(k,\ell)$ and has length $\frac{(1-\epsilon^3)F\epsilon^k}{\epsilon^7} = |I(k,\ell)|/\epsilon^4$. During $I$, the algorithm process jobs of $J_k$ that are either in (1) $Q(i,k,\ell)$, or (2) processed by $\mathcal{A}$ on machine $i$. From Property 2, the total processing of jobs in (2) is $(1-\epsilon^3)|I|$. This leaves us with $\epsilon^3|I|$ processing time. This is enough of process the jobs in $Q(i,k,\ell)$ and $j'$ as $(1-\epsilon^3)\frac{F\epsilon^k}{\epsilon^3} + F\epsilon^{k-1} \leq \frac{F\epsilon^k}{\epsilon^4} = \epsilon^3|I|$. So the flow time of $j$ is at most $|I| + |I(k,\ell)| = F\epsilon^k(\frac{1}{\epsilon^7} + \frac{1}{\epsilon^3})$. ◀

## 5 Removing the assumption about knowledge of $F$

In this section, we show how to remove the assumption about knowledge of $F$. We apply the standard double trick that is often used in the online algorithms. Recall that our previous look-ahead algorithm assumed that we know the optimal $F$. Here, we will construct another look-ahead algorithm $\mathcal{C}$ which will invoke $\mathcal{A}$ for different guesses of $F$. Fix an instance $\mathcal{I}$. Let $I(k,\ell,F)$ be the interval $[\frac{\ell F\epsilon^k}{\epsilon^3}, \frac{(\ell+1)F\epsilon^k}{\epsilon^3})$. This is same as $I(k,\ell)$ except that the intervals are also parameterized by $F$. Similarly, we say that a job of class $k$ is of type $(k,\ell,F)$ if $r_j \in I(k,\ell,F)$.

Our algorithm will work with the guesses of $F$ which are powers of $\left(\frac{1+\epsilon^3}{\epsilon^3}\right)$. Without the loss of generality, we assume that all release dates and processing times are integers. We first generalize the algorithm $\mathcal{A}$. The new algorithm, denoted by $\mathcal{A}'$, will take as parameters an instance $I'$, the guess $F$ and a starting time $t_0$ - all release dates in $I'$ will be at least $t_0$. It will run $\mathcal{A}'$ with the understanding that time start at $t_0$. The interval $I(k,\ell,F)$ will be defined as $[t_0 + \frac{\ell F\epsilon^k}{\epsilon^3}, t_0 + \frac{(\ell+1)F\epsilon^k}{\epsilon^3})$. The algorithm $\mathcal{C}$ is described below.

**Algorithm 5** A look-ahead algorithm $\mathcal{C}$.

```
1: Initialize F_0 = 1, t_0 = 0, I_0 = I
2: for u = 0, 1, 2, ... do
3:     Run A'(I_u, F_u, t_u)
4:     if All non-rejected jobs are finished then
5:         Stop and output the scheduled produced.
6:     else
7:         let j be the first non-rejected job which algorithm A'(I_u, F_u, t_u) is not to schedule.
8:         Suppose j is of type (k, ℓ, F_u).
9:         Define t_{u+1} be the end-point of I(k, ℓ, F_u).
10:        Define I_{u+1} be the jobs in I_u which are not scheduled yet.
11:        Define r_j = max{t_{u+1}, r_j}, ∀j ∈ I_{u+1}.
12:        Set F_{u+1} = F_u ((1+ε^3)/ε^3).
```

Note that this algorithm, like Algorithm $\mathcal{A}$, is preemptive.

### 5.1 Analysis

Suppose during some iteration $u$, we find a job $j^*$ that the algorithm is not able to schedule in iteration $u$. Let $j^*$ be type of $(k^*,\ell^*,F_u)$. Recall that $t_{u+1}$ is the end point of $I(k^*,\ell^*,F_u)$. For a job $j \in \mathcal{I}_u$ let $r_j^u$ denote its release date in $\mathcal{I}_u$.

▶ **Lemma 11.** *Any job $j \in \mathcal{I}_{u+1}$ with $r_j^u < t_{u+1}$ must be of class at most $k^*$. Further, if such a job is of class $k$, then $t_{u+1} - r_j \leq F_{u+1}\epsilon^k$.*

**Proof.** Suppose $j \in \mathcal{I}_u$ and $r_j^u < t_{u+1}$. If $j$ is of type $(k, \ell, F_u)$ such that $k > k^*$, then $I(k, \ell, F_u) \subseteq I(k^*, \ell^*, F_u)$. Hence the interval $I(k, \ell, F_u)$ end at or or before $t_{u+1}$. By definition of $j^*$ the algorithm must have scheduled $j$ in $I(k, \ell, F_u)$ and so, before the $t_{u+1}$. This proves the first statement in the lemma.

To prove the second statement of lemma, we use induction on $u$. Suppose the second statement is true for iteration $u - 1$. We show that it holds for $u$. Let $j$ be job of class $k \leq k^*$ such that $j \in \mathcal{I}_{u+1}$ and $r_j^u < t_{u+1}$. Note that interval $I(k, \ell, F_u)$ ends on or after $t_{u+1}$. Hence $t_{u+1} - r_j^u \leq |I(k, \ell, F_u)| = \frac{T_u \epsilon^k}{\epsilon^3}$. If $r_j \geq t_u$, then $r_j^u = r_j$, and we have $t_{u+1} - r_j \leq |I(k, \ell, F_u)| = \frac{F_u \epsilon^k}{\epsilon^3} = \frac{F_{u+1} \epsilon^k}{(1+\epsilon^3)} \leq F_{u+1} \epsilon^k$.

On the other hand, if $r_j^u = t_u$. So we get $t_{u+1} - t_u \leq |I(k^*, \ell^*, F_u)| \leq \frac{F_u \epsilon^{k^*}}{\epsilon^3} \leq \frac{F_u \epsilon^k}{\epsilon^3}$. By induction hypothesis, we have $t_u - r_j < F_u \epsilon^k$. Hence we have $t_{u+1} - r_j = t_{u+1} - t_u + t_u - r_j \leq \left(\frac{1+\epsilon^3}{\epsilon^3}\right) F_u \epsilon^k = F_{u+1} \epsilon^k$. ◀

▶ **Lemma 12.** *If $\mathcal{C}$ does not finish all jobs in the iteration $u$, then the value of offline optimal solution is at least $F_u$.*

**Proof.** The proof is similar to Lemma 8. The set $S$ is defined similarly. For each machine and interval $I(k, \ell, F_u)$ the algorithm rejects at least $\epsilon^2$-fraction of volume of jobs. Lemma 4 and Lemma 6 remain unchanged.

Note that for a job $j$ of type $(k, \ell, F_u)$, $r_j^u$ may lie earlier than the start time of $I(k, \ell, F_u)$. So the optimum offline algorithm may complete processing $j$ even before the start of this interval. But Lemma 11 shows that $j$ is released at most $\epsilon^3 |I(k, \ell, F_u)|$ to the left of $I(k, \ell, F_u)$. So in the definition of the intervals $I^\epsilon(k, \ell, F_u)$ in Lemma 4, we consider the interval $I(k, \ell, F_u)$ and two segments of length $\epsilon^3 |I(k, \ell, F_u)|$ both before and after $I(k, \ell, F_u)$. Rest of the arguments are same as in the proof of Lemma 8. ◀

▶ **Corollary 13.** *Suppose $OPT$ lies between $F_{u-1}$ and $F_u$. Then the algorithm $\mathcal{C}$ completes a job of class $k$ with flow-time at most $\frac{(1+\epsilon^3) F_u \epsilon^k}{\epsilon^3}$*

## 5.2 Making the algorithm online

We now describe the final on-line algorithm $\mathcal{D}$. The above theorem implies that for any job $j$, we will know the machine on which it get schedules by time $r_j + \frac{(1+\epsilon^3) F_u \epsilon^k}{\epsilon^3}$. At this time, we place $j$ on the queue of the machine to which it gets scheduled on by $\mathcal{C}$. Further each machine prefer the jobs of larger class and within a particular class, it just goes by processing times. We reject at least $2\epsilon^2$ volume of jobs in each interval. To achieve this, the algorithm rejects job $4\epsilon^2$-jobs ($\epsilon$-fraction as in $\mathcal{A}$ and $3\epsilon$-fraction on each machine $i$) in the description of the algorithm $\mathcal{B}$. Hence Property 2 for the algorithm $\mathcal{B}$ can be changed slightly to show that $|P(i, k, \ell)|$ is at most $\frac{(1-2\epsilon^3) F \epsilon^k}{\epsilon^3}$.

▶ **Theorem 14.** *In the schedule $\mathcal{D}$ a job $j$ of class $k$ has flow-time at most $\frac{F \epsilon^k}{\epsilon^8}$. Hence the algorithm $\mathcal{D}$ is an $O(\frac{1}{\epsilon^9})$-competitive algorithm that rejects at most $O(\epsilon)$-fraction of total weights of job.*

## References

1 C. Ambühl and M. Mastrolilli. On-line scheduling to minimize max flow time: an optimal preemptive algorithm. *Oper. Res. Lett.*, 33(6):597–602, 2005.

2 S. Anand, K. Bringmann, T. Friedrich, N. Garg, and A. Kumar. Minimizing Maximum (Weighted) Flow-time on Related and Unrelated Machines. In *Proceedings of International Colloquium on Automata, Languages and Programming*, pages 13–24, 2013.

3 S. Anand, N. Garg, and A. Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proceedings of Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

4 N. Bansal and E. Cloostermans. Minimizing Maximum Flow-Time on Related Machines. In *Proceedings of Workshop on Approximation Algorithms for Combinatorial Problems*, pages 1–14, 2015.

5 N. Bansal and K. Pruhs. Server Scheduling in the Weighted $\ell$p Norm. In *Proceedings of Latin American Symposium on Theoretical Informatics*, pages 434–443, 2004.

6 Chandra Chekuri, Sungjin Im, and Benjamin Moseley. Online Scheduling to Minimize Maximum Response Time and Maximum Delay Factor. *Theory of Computing*, 8(1):165–195, 2012.

7 A.R. Choudhury, S. Das, N. Garg, and A .Kumar. Rejecting jobs to Minimize Load and Maximum Flow-time. In *Proceedings of Symposium on Discrete Algorithms*, pages 1114–1133, 2015.

8 A.R. Choudhury, S. Das, A. Kumar, P. Harsha, and G. Ramalingam. Minimizing weighted $\ell_p$-norm of flow-time in the rejection model. In *Proceedings on the Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 45, pages 25–37, 2015.

9 P.F. Dutot, E. Saule, A. Srivastav, and Denis D. Trystram. Online Non-preemptive Scheduling to Optimize Max Stretch on a Single Machine. In *Proceedings of nternational Conference on Computing and Combinatorics*, pages 483–495, 2016.

10 K. Fox, S. Im, and B. Moseley. Energy efficient scheduling of parallelizable jobs. In *Proceedings of Symposium on Discrete Algorithms*, pages 948–957, 2013.

11 A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *Proceedings of Symposium on Discrete Algorithms*, pages 1242–1253, 2012.

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

13 Giorgio Lucarelli, Nguyen Kim Thang, Abhinav Srivastav, and Denis Trystram. Online Non-preemptive Scheduling in a Resource Augmentation Model based on Duality. In *European Symposium on Algorithms (ESA, 2016)*, volume 57, pages 1–17, 2016.

14 B.A. Michael, S. Chakrabarti, and S. Muthukrishnan. Flow and Stretch Metrics for Scheduling Continuous Job Streams. In *Proceedings of the Annual Symposium on Discrete Algorithms*, pages 270–279, 1998.

15 C.A. Phillips, C. Stein, and E. Torng andJ. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

16 Im S, B. Moseley, K. Pruhs, and C. Stein. Minimizing Maximum Flow Time on Related Machines via Dynamic Posted Pricing. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 51:1–51:10, 2017.

17 N.K. Thang. Lagrangian Duality in Online Scheduling with Resource Augmentation and Speed Scaling. In *Proceedings of European Symposium on Algorithms*, pages 755–766, 2013.