Online Non-Preemptive Scheduling to Minimize Weighted Flow-time on Unrelated Machines

Giorgio Lucarelli
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, France
giorgio.lucarelli@imag.fr

Benjamin Moseley
Carnegie Mellon University, USA
moseleyb@andrew.cmu.edu

Nguyen Kim Thang
IBISC, Univ Evry, University Paris-Saclay, France
thang@ibisc.fr

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

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

Abstract

In this paper, we consider the online problem of scheduling independent jobs *non-preemptively* so as to minimize the weighted flow-time on a set of unrelated machines. There has been a considerable amount of work on this problem in the preemptive setting where several competitive algorithms are known in the classical competitive model. However, the problem in the non-preemptive setting admits a strong lower bound. Recently, Lucarelli et al. presented an algorithm that achieves a $O\left(\frac{1}{\epsilon^2}\right)$-competitive ratio when the algorithm is allowed to reject ϵ-fraction of total weight of jobs and has an ϵ-speed augmentation. They further showed that speed augmentation alone is insufficient to derive any competitive algorithm. An intriguing open question is whether there exists a scalable competitive algorithm that rejects a small fraction of total weights.

In this paper, we affirmatively answer this question. Specifically, we show that there exists a $O\left(\frac{1}{\epsilon^3}\right)$-competitive algorithm for minimizing weighted flow-time on a set of unrelated machine that rejects at most $O(\epsilon)$-fraction of total weight of jobs. The design and analysis of the algorithm is based on the primal-dual technique. Our result asserts that alternative models beyond speed augmentation should be explored when designing online schedulers in the non-preemptive setting in an effort to find provably good algorithms.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Online Algorithms, Scheduling, Resource Augmentation

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.59

Funding Nguyen Kim Thang is supported by the ANR project OATA n° ANR-15-CE40-0015-01, Hadamard PGMO and DIM RFSI. Abhinav Srivastav was supported by the PSL project Multi-Fac and is supported by the ANR project OATA n° ANR-15-CE40-0015-01, Hadamard PGMO and DIM RFSI. Benjamin Moseley was supported in part by a Google Research Award, and NSF Grants CCF-1617724, CCF-1733873 and CCF-1725661.

1 Introduction

In this work, we study the fundamental problem of online scheduling of independent jobs on unrelated machines. Jobs arrive over time and the online algorithm has to make the decision which job to process *non-preemptively* at any time on each machine. A job j is released at time r_j and takes p_{ij} amount of processing time on a machine i. Further, each job has a weight w_j that denotes its (relative) priority. Our aim is to design a non-preemptive schedule that minimizes the total weighted flow-time (or response time) quantity, i.e., $\sum_j w_j(C_j - r_j)$ where C_j denotes the completion time of job j.

We are interested in designing online non-preemptive scheduling problem in the worst-case model. Several strong lower bounds are known for simple instances [2, 4]. The main hurdle arises from two facts: the algorithm must be online and robust to all problem instances and the algorithmic decisions made should be of irrevocable nature. In order to overcome the strong theoretical lower bound, Kalyanasundaram and Pruhs [7] and Phillips et al. [10] proposed the analysis of scheduling algorithms in terms of the speed augmentation and machine augmentation, respectively. Together these augmentation are commonly referred to as resource augmentation. Here, the idea is to either give the scheduling algorithm faster processors or extra machines in comparison to the adversary. For preemptive problems, these models provide a tool to establish a theoretical explanation for the good performance of algorithms in practice. In fact, many practical heuristics have been shown to be competitive where the algorithm is given resource augmentation. In contrast, problems in the non-preemptive setting have resisted against provably good algorithms even with such additional resources [9].

Choudhury et al. [5] proposed a new model of resource augmentation where the online algorithm is allowed to reject some of the arriving jobs, while the adversary must complete all jobs. Using a combination of speed augmentation and rejection, Lucarelli et al. [9] break this theoretical barrier and gave a scalable algorithm for non-preemptive weighted flow-time problems. However, it remains an intriguing question about the power of rejection model in comparison to the previous ones.

Recently, Lucarelli et al. [8] showed that a $O(1)$ competitive algorithm exists if all jobs have unit weight and one only rejects a constant fraction of the jobs. Their algorithm and analysis are closely tied to the unweighted case and there is no natural extension to the case where jobs have weights. The question looms, does there exist a constant competitive algorithm for non-preemptive scheduling to minimize weighted flow-time using rejection?

1.1 Our Result and Approach

This paper gives the first algorithm with non-trivial guarantees for minimizing weighted flow time using rejection and no other form of resource augmentation. The main result of the paper is the following theorem. The theorem shows that constant competitiveness can be achieved by only rejecting a small faction of the total weight of the jobs.

▶ **Theorem 1.** *For the non-preemptive problem of minimizing weighted flow-time on unrelated machines, there exists a $O(\frac{1}{\epsilon^3})$-competitive algorithm that rejects at most $O(\epsilon)$-fraction of total weight of the jobs for any $0 < \epsilon < 1$.*

The algorithmic decisions are classified into three parts: dispatching, rejecting and scheduling policy. The scheduling follows HDF policy (Highest Density First) once jobs are assigned to the machines. At the arrival of a job, for each machine, the algorithm computes an approximate increase in the weighted flow-time and assigns the job to the machine with

the least increase in the approximate weighted flow-time. To compute this quantity for a given machine, the algorithm considers the set of uncompleted jobs in the machine queue in the non-increasing order of densities and uses two different rejection policies.

The first rejection policy, referred as the *preempt* rule, rejects jobs that have already started processing if the total weight of newly arrived "high priority" jobs (high density jobs) exceeds a given threshold. Specifically, when a job starts executing, we associate a counter that keeps tracks of the total weight of newly arrived jobs. Once the value of this counter is at least $1/\epsilon$ times the weight of the current executing job, the algorithm preempts the current executing job and rejects it. The rejected job is pushed out of the system so as to be never executed again.

We emphasize here a critical issue due to job rejection which is of different nature to speed augmentation. Observe that rejecting a job that has already started processing may cause a large decrease in the weighted flow-time of the jobs in the machine queue. Due to job arrivals and job rejections, quantities associated to the machine queue (for example the remaining job weight, etc) vary arbitrarily without any nice properties like monotonicity. That creates a significant challenge in the dual fitting analysis. To tackle this problem, we introduce the notion of *definitive completion time* for each job. Once a job is rejected or completed before its definitive completion time, the algorithm removes the job from the queue of the machine. However, for the purpose of analysis, the rejected jobs are still considered in the definition of dual variables until their definitive completion time. This ensures that for any fixed time, the weight of jobs not yet definitively completed increases with the arrival of new jobs (see Section 3.4 for details).

The second rejection policy, referred as the *weight-gap* rule, rejects unprocessed "low priority" jobs (small density jobs) from the machine's queue. This policy simulates the ϵ-speed-augmentation. In the particular case where all jobs have the same weight, this rejection policy rejects a "low priority" job for every $1/\epsilon$ arrivals of new jobs. Due to the scheduling policy, if a "low priority" job is not rejected, then it will be completed last in the schedule (assuming no future job arrivals).

In the algorithm's schedule, future arriving jobs do not delay the rejected low priority jobs, while the later ones need to be completed in the adversary's schedule. This is where the algorithm benefits from the power of rejection. Specifically, the algorithm can use the difference between the rejection time and the definitive completion time of jobs to create a similar effect to speed augmentation. The key idea is to reject the low priority jobs so their total weight is comparable to jobs that arrive after them.

The definitive completion times play a crucial role so that the dual achieves a substantial value compared to the primal. By carefully choosing the definitive completion times of jobs, we manage to prove the competitive ratio of our algorithm with admittedly sophisticated analysis.

1.2 Related Works

The problem of minimizing the total weighted flow-time has been extensively studied in the online scenario. For the preemptive problem, Chekuri et al. [4] presented a *semi-online* $O(\log^2 P)$-competitive algorithm for a single machine, where P is the ratio of the largest to the smallest processing time of the instance. Later, Bansal and Dhamdhere [3] proposed a $O(\log W)$-competitive algorithm, where W is the ratio between the maximum and the minimum weights of the jobs. This was later improved in [2]. In contrast to the single-machine case, Chekuri et al. [4] showed a $\Omega(\min(\sqrt{P}, \sqrt{W}, \frac{n}{m}^{\frac{1}{4}}))$ lower bound for m identical machines. For the online non-preemptive problem of minimizing the total weighted flow-time, Chekuri et al. [4] showed that any algorithm has at least $\Omega(n)$ competitive ratio for single machine where n is the number of jobs.

In speed-augmentation model, Anand et al. [1] presented a scalable competitive algorithm for the preemptive problem on a set of unrelated machines. For the non-preemptive setting, Phillips et al. [10] gave a constant competitive algorithm in identical machine setting that uses $m \log P$ machines (recall that the adversary uses m machines). They also showed that there exists a $O(\log n)$-machine $O(1)$-speed algorithm that returns the optimal schedule for the unweighted flow-time objective. Epstein et al. [6] proposed an ℓ-machines $O(\min\{\sqrt[\ell]{P}, \sqrt[\ell]{n}\})$-competitive algorithm for the unweighted case on a single machine. This algorithm is optimal up to a constant factor for constant ℓ.

Lucarelli et al. [9] presented a strong lower bound on the competitiveness for the weighted flow-time problem on a single machine that uses arbitrarily faster machine than that of the adversary. Choudhury et al. [5] extended the resource augmentation model to allow *rejection*, according to which the algorithm does not need to complete all jobs and some of them can be rejected. Using a combination of speed augmentation and rejection, Lucarelli et al. [9] gave a constant competitive algorithm for the weighted flow-time problem on a set of unrelated machines. In particular, they showed that there exists a $O(1/\epsilon^2)$-competitive algorithm that uses machines with speed $(1+\epsilon)$ and rejects at most an ϵ-fraction of jobs for arbitrarily small $\epsilon > 0$. Recently, Lucarelli et al. [8] provided a scalable competitive algorithm for the case of (unweighted) flow time where there is no speed augmentation.

2 Definitions and Notations

2.1 Problem definition

We are given a set $\mathcal{M}$ of unrelated machines and a set of jobs $\mathcal{J}$ that arrive online. Each job j is characterized by its release time r_j and its weight w_j. If job j is executed on machine i, it has a processing requirement of p_{ij} time units. The goal is to schedule jobs *non-preemptively*. Given a schedule $\mathcal{S}$, the *completion time* of the job j is denoted by $C_j^{\mathcal{S}}$. The *flow-time* of j is defined as $F_j^{\mathcal{S}} = C_j^{\mathcal{S}} - r_j$, which is the total amount of time job j remains in the system. The objective is to minimize the weighted flow-times of all jobs, i.e., $\sum_{j\in\mathcal{J}} w_j F_j^{\mathcal{S}}$. In the following section we formulate this problem as a linear program.

2.2 Linear Programming Formulation

The LP formulation presented below is an extension of those used in the prior works of [1, 9]. For each job j, machine i and time $t \geq r_j$, there is a binary variable x_{ijt} which indicates if j is processed or not on i at time t. The problem of minimizing weighted flow-time can be expressed as:

$$\min \sum_{i,j,t} w_j \left(\frac{t - r_j}{p_{ij}} + 21 \right) x_{ijt}$$

$$\sum_{i,t} \frac{x_{ijt}}{p_{ij}} = 1 \qquad \forall j \tag{1}$$

$$\sum_{j} x_{ijt} \leq 1 \qquad \forall i, t \tag{2}$$

$$x_{ijt} \in \{0,1\} \qquad \forall i, j, t \geq r_j \tag{3}$$

The objective value of the above integer program is at most a constant factor than that of the optimal preemptive schedule. The above integer program can be relaxed to a linear

program by replacing the integrality constraints of x_{ijt} with $0 \leq x_{ijt} \leq 1$. The dual of the relaxed linear program can be expressed as follows:

$$\max \sum_j \alpha_j - \sum_{i,t} \beta_{it}$$

$$\frac{\alpha_j}{p_{ij}} - \beta_{it} \leq w_j \left(\frac{t - r_j}{p_{ij}} + 21 \right) \qquad \forall i, j, t \geq r_j \tag{4}$$

$$\beta_{it} \geq 0 \qquad \forall i, t \tag{5}$$

For the *rejection model* considered in this work, it is assumed that the algorithm is allowed to reject jobs. Rejection can be interpreted in the primal LP by only considering constraints corresponding to non-rejected jobs. That is, the algorithm does not have to satisfy the constraint (1) for rejected jobs.

2.3 Notations

In this section, we define notations that will be helpful during the design and analysis of the algorithm.

- t^- denotes the time just before t that is, $t^- = t - \epsilon'$ for an arbitrarily small value of $\epsilon' > 0$.
- $U_i(t)$ denotes the set of pending jobs at time t on machine i, i.e., the set of jobs dispatched to i that have not yet completed and also have not been rejected until t.
- $\kappa_i(t)$ denotes the job currently executing on machine i at time t.
- $V_i(t)$ denotes the set of unprocessed jobs in $U_i(t)$ that is $V_i(t) = U_i(t) \backslash \{\kappa_i(t)\}$. Throughout this paper, we assume that the jobs in $V_i(t)$ are indexed in non-increasing order of their densities that $\delta_{i1} \geq \delta_{i2}, \ldots, \geq \delta_{i|V_i(t)|}$.
- $\nu_i(t)$ denotes the smallest density job in $V_i(t)$.
- $R_i^1(a, b)$ denotes the set of jobs rejected due to the *prempt* rule (to be defined later) during time interval $(a, b]$. In particular, $R_i^1(t)$ is the set of job rejected at time t due to the *prempt rule*.
- Similarly, $R_i^2(a, b)$ denotes the set of jobs rejected due to the *weight-gap* rule (also to be defined later) during time interval $(a, b]$. In particular, $R_i^2(t)$ is the set of job rejected at time t due to the weight-gap rule.
- $q_{ij}(t)$ denotes the remaining processing time of j at a time t on machine i.
- δ_{ij} is the density of a job j on machine i that is $\delta_{ij} = \frac{w_i}{p_{ij}}$.
- S_j denotes the starting of job j on some machine i. If a job is rejected before it starts executing, set $S_j = \infty$.

By the previous definitions, it follows that $R_i^1(r_j), R_i^2(r_j) \subseteq U(r_j^-) \cup \{j\}$ and $U(r_j) = (U(r_j^-) \cup \{j\}) \setminus \{R_i^1(r_j) \cup R_i^2(r_j)\}$.

3 The Algorithm

In this section, we describe our algorithm. Specifically, we explain how to take the following decisions: *dispatching* that is to decide the machine assignment of jobs; *scheduling* that is to decide which jobs to process at each time; and *rejection*. The algorithm is denoted by $\mathcal{A}$. Let $0 < \epsilon < 1$ be an arbitrarily small constant. Note that the proposed algorithm rejects an $O(\epsilon)$-fraction of the total weight of jobs and dispatches each job to a machine upon its arrival.

3.1 Scheduling policy

At each time t if the machine i is *idle* either due to the rejection of a job or due to the completion of a job, then the algorithm starts executing the job j with the highest density among all the jobs in $U_i(t)$, *i.e.* $j = \arg\max_{h \in U_i(t)} \delta_{ih}$. In case of ties, the algorithm selects the job with the earliest release time.

3.2 Rejection policies

Our algorithm uses two different rules for rejecting jobs. The first rule called as the *preempt rule*, bounds the total weight of "high priority" jobs that arrive during the execution of a job. The second rule called as the *weight-gap* rule, helps the algorithm to balance the total amount of weight of low density jobs. The algorithm associates two counters, count^1_j and count^1_j, with each job j which are both initialized to 0 at r_j.

1. **Preempt rule**: Let $j = \kappa_i(t)$ be the job processing on i at time t. During the processing of j, if a new job j' is dispatched to i then count^1_j is incremented by $w_{j'}$. Let k be the earliest job released and dispatched to machine i during the execution of j such that $\mathsf{count}^1_j \geq w_j/\epsilon$, if it exists. At r_k, the algorithm interrupts the processing of j and rejects it, that is $R^1_i(r_k) = \{j\}$. If no job is rejected due to the *preempt* rule at r_k, then we set $R^1_i(r_k) = \emptyset$.
2. **Weight-gap rule**: We associate a function $W_i(t) : \mathbb{R}^+ \to \mathbb{R}^+$ with each machine i which is initialized to 0 for every t. Informally $W_i(t)$ represents the total budget for future rejections. If a job j is dispatched to machine i then $W_i(t)$ for $t \geq r_j$ is updated according to the following policy.
   Let $V = V_i(r_j^-) \cup \{j\}$. Assume that the jobs in V are indexed in non-increasing order of their densities that is, $\delta_{i1} \geq \delta_{i2} \geq \ldots \geq \delta_{i\nu}$, where the job with index ν is the smallest density job in V. Note that the job j is included in this ordering. Let s be the smallest index in $\{1, 2, \ldots, \nu\}$ such that:

$$\sum_{h=s}^{\nu} w_h \leq \epsilon(W_i(t^-) + w_j) < \sum_{h=(s-1)}^{\nu} w_h \tag{6}$$

   We say that no such job with index s exists if and only if $w_\nu > \epsilon(W_i(t^-)+w_j)$. Algorithm 1 defines the set of jobs $R^2_i(r_j)$. The algorithm rejects the jobs in $R^2_i(r_j)$ and updates $W_i(t)$ as follows:

$$W_i(t) = \max\{0, W_i(r_j^-) + w_j - \sum_{h \in R^2_i(r_j)} w_h/\epsilon\}, \qquad \forall t \geq r_j \tag{7}$$

The following lemma describes some properties arising due to the *weight-gap* rule.

▶ **Lemma 2.** *The following properties hold.*
(Property 1) *If $R^2_i(r_j) = \{\nu_i(r_j^-), j\}$ or $R^2_i(r_j) = \{(s-1), \ldots, v\}$ then $W_i(r_j) = 0$.*
(Property 2) *$\epsilon W_i(t) < w_{\nu_i(t)}$ for every pair of i, t.*
(Property 3) *Let $w_{|R^2_i(r_j)|}$ denote the weight of smallest density job in $R^2_i(r_j)$. If $j \notin R^2_i(r_j)$ then $\sum_{h \in R^2_i(r_j)} w_h - w_{|R^2_i(r_j)|} \leq 2\epsilon w_j$.*
(Property 4) *If $j \in R^2_i(r_j)$, then $R^2_i(r_j) = \{j\}$ or $\{j, \nu_i(r_j^-)\}$.*

▶ **Lemma 3.** *The total weight of jobs rejected by the* preempt rule *is at most $O(\epsilon)$-fraction of the total weight of jobs in $\mathcal{J}$.*

Algorithm 1 Weight-gap Rejection Rule.

if no job with index s exists **then**
  if j is not the smallest density job in V **then**
    No job is rejected that is, $R_i^2(r_j) := \emptyset$
  else
    $\{j$ is the smallest density job in V that is, j is the job with index $\nu\}$
    $\{v_i(r_j^-)$ is the job with index $\nu - 1\}$
    if $p_{ij} \geq \epsilon p_{i(\nu-1)}$ **then**
      No job is rejected that is, $R_i^2(r_j) := \emptyset$
    else
      $\mathsf{count}^2_{(\nu-1)} := \mathsf{count}^2_{(\nu-1)} + w_j$
      if $\mathsf{count}^2_{(\nu-1)} \geq w_{(\nu-1)}$ **then**
        Reject j and $\nu_i(r_j^-)$ that is, $R_i^2(r_j) := \{j, \nu_i(r_j^-)\}$
      else
        No job is rejected that is, $R_i^2(r_j) := \emptyset$
else
  {a job with index s exists}
  if $w_j \geq w_{(s-1)}/\epsilon$ **then**
    Reject jobs with indices $s-1, \ldots, \nu$ in V that is, $R_i^2(r_j) := \{s-1, \ldots, \nu\}$
  else
    $\{w_j < w_{(s-1)}/\epsilon\}$
    if j is not one of the jobs in $s, \ldots, \nu$ that is, $j \notin \{s, \ldots, \nu\}$ **then**
      Reject jobs with indices $s, \ldots, \nu$ in V that is, $R_i^2(r_j) := \{s, \ldots, \nu\}$
    else
      $\{j \in \{s, \ldots, \nu\}\}$
      $\mathsf{count}^2_{(s-1)} := \mathsf{count}^2_{(s-1)} + w_j$
      if $\mathsf{count}^2_{(s-1)} \geq w_{(s-1)}$ **then**
        Reject jobs with indices $s-1, \ldots, \nu$ in V that is, $R_i^2(r_j) := \{s-1, \ldots, \nu\}$.
      else
        Reject jobs with indices $s, \ldots, \nu$ in V that is, $R_i^2(r_j) := \{s, \ldots, \nu\}$.

Proof. From *preempt* rule, it follows that each job j can be associated with a set of jobs such that their total weight is at most w_j/ϵ. For every pair of j, j' and $j \neq j'$, the intersection of the associated sets is empty and hence the lemma follows. ◀

▶ **Lemma 4.** *The total weight of jobs rejected by the* weight-gap *rule is at most $O(\epsilon)$-fraction of the total weight jobs in $\mathcal{J}$.*

3.3 Dispatching policy

When a new job j arrives, a variable Δ_{ij} is set. Intuitively, Δ_{ij} is the approximate increase in the total weighted flow-time objective if the job j is assigned to the machine i and j is not rejected. Then, Δ_{ij} is defined as follows.

$$\Delta_{ij} = w_j \sum_{h \in V_i(r_j): \delta_{ih} \geq \delta_{ij}} p_{ih} + p_{ij} \sum_{h \in V_i(r_j): \delta_{ih} < \delta_{ij}} w_h + w_j q_{i\kappa_i(r_j^-)}(r_j) \cdot \mathbb{1}_{\{\kappa_i(r_j^-) \text{ is not rejected currently due to preempt rule}\}}$$

$$- q_{i\kappa_i(r_j^-)}(r_j) \cdot \sum_{h \in U_i(r_j) \setminus \{j\}} w_h \cdot \mathbb{1}_{\{\kappa_i(r_j^-) \text{ is currently rejected due to the preempt rule}\}}$$

The first term corresponds to the flow-time of the new job j due to waiting on jobs with higher density than δ_{ij} in $V_i(r_j)$. The second term corresponds to the delay of the jobs in $V_i(r_j)$ with smaller density than δ_{ij}. The third and the fourth terms give corrections depending on whether job $\kappa_i(r_j^-)$ is rejected due to the *preempt* rule.

We now describe the dispatching policy of jobs to machines. At the arrival time of a job j, we hypothetically assign j to every machine i and compute the variables α_{ij}. Finally, we assign j to the machine that minimizes α_{ij}. For notional purposes, we put an additional apostrophe to previously defined variables. The additional apostrophe stands for the fact that these variables correspond to the case where we hypothetically assign j to i. For example, $R_i^{2'}(r_j)$ denote the set of rejected jobs due to the *weight-gap rule* when j is hypothetically assigned to i. Similarly, $W_i'(r_j)$ denote the function W_i at r_j in the case if j is assigned to i. Further, let $\rho = \rho_{ij}$ be an index of a job in $V_i(r_j^-)$ such that the following two inequalities hold simultaneously:

$$\sum_{h=\rho}^{|V_i(r_j^-)|} w_h \leq W_i'(r_j) < \sum_{h=(\rho-1)}^{|V_i(r_j^-)|} w_h$$

The variable α_{ij} is computed for each machine i as follows:

$$\alpha_{ij} = \frac{20 w_j p_{ij}}{\epsilon} + w_j \sum_{h \in V_i(r_j^-) : \delta_{ih} \geq \delta_{ij}} p_{ih} + w_j p_{ij} + p_{ij} \sum_{h \in V_i(r_j^-) : \delta_{ij} > \delta_{ih}} w_{ih} - n_{ij}$$

where n_{ij} is defined as follows.

$$n_{ij} = w_j \left(\sum_{h \in V_i(r_j^-) : \delta_{i\rho} \geq \delta_{ih}} p_h + \left(W_i'(r_j) - \sum_{h \in V_i(r_j^-) : \delta_{i\rho} \geq \delta_{ih}} w_h \right) \frac{p_{i,(\rho-1)}}{w_{(\rho-1)}} \right) \qquad \text{if } R_i^{2'}(r_j) = \{j\},$$

$$n_{ij} = w_j \sum_{h \in R_i'^2(r_j)} p_{ih} \qquad \text{if } R_i'^2(r_j) = \{j, \nu_i(r_j^-)\},$$

$$n_{ij} = p_{ij} \sum_{h \in R_i'^2(r_j)} w_h + \epsilon^2 W_i'(r_j) p_{ij} \qquad \text{otherwise.}$$

The algorithm assigns j to machine $i^* = \arg\min_{i \in \mathcal{M}} \alpha_{ij}$.

3.4 Dual variables

Suppose job j is assigned to machine i. Assume L_j represents the last time t such that j is in $U_i(t)$. Informally, L_j is the time at which j is removed from the queue of the machine i. Note that j can be removed from $U_i(t)$ due to three following reasons:

1. If j has being scheduled for p_{ij} time units on machine i then $L_j = C_j$
2. If j is rejected due to *preempt* rule
3. If j is rejected due to *weight-gap* rule.

In cases 2 and 3, j is rejected due to the arrival of some job, denoted by $\mathsf{rej}(j)$. Recall that $R_i^1(r_j, L_j)$ is the set of jobs that are rejected due to *preempt rule* during the interval $(r_j, L_j]$ on machine i. Note that those jobs cause a decrease in the flow of j. Observe that $R_i^1(r_j, L_j)$ contains j if j is rejected due to the *preempt* rule. We define the *definitive completion* time, denoted by $\widetilde{C}_j$, of a job j as follows.

1. If j is not rejected due to the *weight-gap* rule (corresponds to cases 1 and 2) .

$$\widetilde{C}_j = L_j + \sum_{h \in R_i^1(r_j, L_j)} q_{ih}(r_{\mathsf{rej}(h)}) \tag{8}$$

2. If j is rejected due to the *weight-gap* rule on the arrival of some job other than j that is, $r_{j'}$ where $j' \neq j$

$$\widetilde{C}_j = L_j + \sum_{h \in R_i^1(r_j, L_j)} q_{ih}(r_{\mathsf{rej}(h)}) + \sum_{h \in U_i(L_j): \delta_{ih} \geq \delta_{ij}} q_{ih}(L_j) + \sum_{h \in R_i^2(r_{\mathsf{rej}(j)}): \delta_{ih} \geq \delta_{ij}} p_{ih} \tag{9}$$

3. If j is immediately rejected (i.e., $j \in R_i^2(r_j)$) and job $\nu_i(r_j^-)$ is also rejected due to the arrival of j.

$$\widetilde{C}_j = L_j + p_{ij} + \sum_{h \in U_i(L_j)} q_{ih}(L_j) \tag{10}$$

4. If j is immediately rejected and it is the only job rejected due to the *weight gap* rule at r_j. Denote $\rho = \rho_{ij}$.

$$\widetilde{C}_j = L_j + p_{ij} + \sum_{h \in V_i(L_j^-): \delta_{ih} > \delta_{i(\rho-1)}} p_{ih} + \left(1 - \frac{W_i(L_j) - \sum_{h \in V_i(L_j^-): \delta_{i\rho} \geq \delta_{ih}} w_h}{w_{(\rho-1)}}\right) p_{i(\rho-1)} + q_{i\kappa_i(L_j)}(L_j).\mathbb{1}_{\{R_i^1(L_j)=\emptyset\}} \tag{11}$$

This completes the description of the *definitive completion time*.

Let $Q_i(t)$ denote the set of jobs that have not been definitely completed that is

$Q_i(t) := \{j : j \text{ has been assigned to } i, t < \widetilde{C}_j\}$.

Next, we define the notion of *artificial fractional weight* of a job $j \in Q_i(t)$,

$$w_j^f(t) = \begin{cases} w_j & \text{if } r_j \leq t \leq \widetilde{C}_j - p_{ij} \\ w_j \left(\frac{\widetilde{C}_j - t}{p_{ij}}\right) & \text{if } \widetilde{C}_j - p_{ij} < t < \widetilde{C}_j \end{cases}$$

Now, we have all the necessary tools to set dual variables. At the arrival of job j, set

$$\alpha_j = \left(\frac{\epsilon}{1+\epsilon}\right) \min_{i \in \mathcal{M}} \alpha_{ij}$$

and never change this value again. The second dual variable β_{it} is set to

$$\frac{\epsilon}{(1+\epsilon)(1+\epsilon^2)} \sum_{h \in Q_i(t)} w_h^f(t)$$

Let $Q_i^R(t) \subseteq Q_i(t)$ be the set of jobs that are rejected due to the *weight-gap rule* and are not yet definitively completed until time t.

▶ **Lemma 5.** *For fixed time t, β_{it} may only increase as new jobs arrive and some old jobs might get rejected.*

Observe that above lemma holds as jobs are removed from $Q_i(t)$ only after their *definitive completion time*. Thus a job that might have already completed its execution on a machine or rejected, can still be present in the $Q_i(t)$. During the analysis, we will show that the dual constraint corresponding to job j are feasible at r_j. Since β_{it} only increases with respect to the arrival of new jobs, the feasibility holds for all $t \geq r_j$.

4 Analysis

We present first two technical lemmas which are important for the analysis of our primal-dual algorithm. In Lemma 6, we relate the weight of rejected jobs in $Q_i^R(t)$ to the weight of jobs pending in $U_i(t)$. This will help us in proving the feasibility of dual constraints in Lemma 9, Lemma 10 and Lemma 11. In Lemma 7, we show that the negative parts in the definition of α_js' are relatively small. This will help us to bound the value of the dual objective.

▶ **Lemma 6.** *Let $\kappa = \kappa_i(t)$. For any machine i and any time t, it holds that $\frac{w_\kappa}{p_{i\kappa}} q_{i\kappa}(t) + \sum_{h \in V_i(t)} w_h^f(t) - W_i(t) \leq \frac{1}{\epsilon} \sum_{h \in Q_i^R(t)} w_h(t)$.*

Proof. We prove by induction on the arrival of jobs. The base case when no job has been released, holds trivially. Assume that the above inequality holds for every time t, before the arrival of job j on machine i, we show that it holds after j arrives. We split the proof into two cases depending upon if j is immediately rejected or not. The rest of the proof is omitted due to space constraints. ◀

▶ **Lemma 7.** *Let $J_i(t)$ denote the set of jobs dispatched to machine i until the time t that is, $J(i) = \bigcup_{t' \leq t} U_i(t')$. Then the following inequality holds at all time and for all $i \in \mathcal{M}$*

$$\mathcal{D}^1 - \mathcal{D}^2 \leq \mathcal{B}^1 + \mathcal{B}^2 + \mathcal{B}^3 \tag{12}$$

where

$$\mathcal{D}^1 = \sum_{j \in J_i(t) \setminus R_i^2(r_j)} \left(\epsilon^2 W_i(r_j) p_{ij} - w_j p_{i,\nu_i(r_j^-)} . \mathbb{1}_{\{j = \nu_i(r_j) \text{ and } p_j < \epsilon p_{i\nu_i(r_j^-)}\}} \right),$$

$$\mathcal{D}^2 = \sum_{j \in R_i^2(r_j)} \left(w_j p_{i,\nu_i(r_j)} . \mathbb{1}_{\{|R_i^2(r_j)|=1\}} + w_{\nu_i(r_j^-)} p_{i,\nu_i(r_j^-)} . \mathbb{1}_{\{|R_i^2(r_j)|>1\}} \right),$$

$$\mathcal{B}^1 = \sum_{j \in R_i^2(0,t)} w_j p_{ij}, \quad \mathcal{B}^2 = \sum_{j \in J_i(t) \setminus \{R_i^2(0,t) \cup U_i(t)\}} w_j p_{ij} + \epsilon W_i(t) p_{i,\nu_i(t)} \text{ and}$$

$$\mathcal{B}^3 = \sum_{j \in J_i(t)} w_j p_{ij} / \epsilon.$$

Proof. The proof is omitted due to space constraints. ◀

▶ **Corollary 8.** *Let $J_i \subseteq \mathcal{J}$ be the set of jobs dispatched to machine i that $J_i = \bigcup_{t \geq 0} U_i(t)$. Then the following inequality holds for every machine $i \in \mathcal{M}$,*

$$\sum_{j \in \mathcal{J}_i \setminus R_i^2(r_j)} \epsilon^2 W_i(r_j) p_{ij} \leq \left(\frac{5}{\epsilon} \right) \sum_{j \in \mathcal{J}_i} w_j p_{ij}$$

Proof. From Lemma 7, it immediately follows that

$$\sum_{j\in J_i\setminus R_i^2(r_j)}(\epsilon^2 W_i(r_j)p_{ij} - w_j p_{i,\nu_i(r_j^-)}.\mathbb{1}_{\{j=\nu(r_j) \text{ and } p_j<\epsilon p_{i,\nu_i(r_j^-)}\}})$$
$$-\sum_{j\in R_i^2(r_j)}\left(w_j p_{i,\nu_i(r_j)}.\mathbb{1}_{\{|R_i^2(r_j)|=1\}} + w_{\nu_i(r_j^-)}p_{i,\nu_i(r_j^-)}.\mathbb{1}_{\{|R_i^2(r_j)|>1\}}\right) \le \frac{2}{\epsilon}\left(\sum_{j\in J_i} w_j p_{ij}\right)$$

Rearranging the terms, we get

$$\epsilon^2\sum_{j\in J_i\setminus R_i^2(r_j)} W_i(r_j)p_{ij}$$
$$-\sum_{j\in R_i^2(r_j)}\left(w_j p_{i,\nu_i(r_j)}.\mathbb{1}_{\{|R_i^2(r_j)|=1\}} + w_{\nu_i(r_j^-)}p_{i,\nu_i(r_j^-)}.\mathbb{1}_{\{|R_i^2(r_j)|>1\}}\right)$$
$$\le \frac{2}{\epsilon}\left(\sum_{j\in J_i} w_j p_{ij}\right) + \sum_{j\in J_i\setminus R_i^2(r_j)} w_j p_{i,v_i(r_j^-)}.\mathbb{1}_{\{j=v(r_j) \text{ and } p_j<\epsilon p_{iv_i(r_j^-)}\}}$$
$$\le \frac{2}{\epsilon}\left(\sum_{j\in J_i} w_j p_{ij}\right) + \sum_{h\in J_i} p_{ih}\sum_{j\in J_i: j=\nu(r_j), h=\nu(r_j^-)} w_j$$
$$\le \frac{2}{\epsilon}\left(\sum_{j\in J_i} w_j p_{ij}\right) + \sum_{h\in J_i} p_{ih}w_h/\epsilon$$

since $\mathsf{count}_h^2 < w_h/\epsilon$, otherwise h is rejected due to Line 12 in Algorithm 1

$$\le \frac{3}{\epsilon}\left(\sum_{j\in J_i} w_j p_{ij}\right)$$

Rearranging the terms again, we get

$$\epsilon^2\sum_{j\in J_i\setminus R_i^2(r_j)} W_i(r_j)p_{ij}$$
$$\le \frac{3}{\epsilon}\left(\sum_{j\in J_i} w_j p_j\right) + \sum_{j\in R_i^2(r_j)}\left(w_j p_{i,\nu_i(r_j)}.\mathbb{1}_{\{|R_i^2(r_j)|=1\}} + w_{\nu_i(r_j^-)}p_{i,\nu_i(r_j^-)}.\mathbb{1}_{\{|R_i^2(r_j)|>1\}}\right)$$
$$\le \frac{4}{\epsilon}\left(\sum_{j\in J_i} w_j p_j\right) + \sum_{j\in R_i^2(r_j)}\left(w_j p_{i,\nu_i(r_j)}.\mathbb{1}_{\{|R_i^2(r_j)|=1\}}\right) \quad \text{since } R_i^2(r_j) = \{j, \nu_i(r_j^-)\}$$
$$\le \frac{4}{\epsilon}\left(\sum_{j\in J_i} w_j p_j\right) + \sum_{h\in J_i} p_{ih}\sum_{j\in J_i: h=\nu_i(r_j^-)=\nu_i(r_j)} w_j$$
$$\le \frac{5}{\epsilon}\left(\sum_{j\in J_i} w_j p_j\right)$$

The last inequality holds since $\mathsf{count}_h^2 < w_h/\epsilon$, otherwise h is rejected in Line 27 in Algorithm 1. Thus, the corollary follows. ◀

Now we show the proof of dual feasibility for each job j on every pair of i, t. Thus, for a given machine i, j may or may not be assigned to i. by the algorithm.

▶ **Lemma 9.** *Suppose that a job j is not immediately rejected at r_j when j is hypothetically assigned to i. Then, the dual constraint (4) corresponding to j holds.*

▶ **Lemma 10.** *Assume that a job j is immediately rejected at r_j and $R_i^2(r_j) = \{j, \nu_i(r_j^-)\}$ when j is hypothetically assigned to i. Then, the dual constraint (4) corresponding to j holds.*

▶ **Lemma 11.** *Suppose that a job j is immediately rejected at r_j and $R_i^2(r_j) = \{j\}$ when j is hypothetically assigned to i. Then, the dual constraint (4) corresponding to j holds.*

▶ **Lemma 12.** *It holds that $\sum_{j \in \mathcal{J}} \alpha_j \geq \frac{\epsilon}{1+\epsilon} \sum_{j \in \mathcal{J}} (\widetilde{C}_j - r_j)$.*

The proofs of above lemmas are omitted due to space constraints

4.1 Proof of theorem 1

Proof. In the definition of dual variables, each job j is accounted in β_{it} variable until its *definitive completion time*. Thus, $\sum_{i,t} \beta_{it} \leq \frac{\epsilon}{(1+\epsilon)(1+\epsilon^2)} \sum_{j \in \mathcal{J}} w_j(\widetilde{C}_j - r_j)$. Combining it with Lemma 12, we have that the dual objective is at least $\frac{\epsilon^3}{(1+\epsilon)(1+\epsilon^2)} \sum_{j \in \mathcal{J}} w_j(\widetilde{C}_j - r_j)$. Further, the cost of the primal is at most $22 \sum_{j \in \mathcal{J}} w_j(\widetilde{C}_j - r_j)$. Hence the theorem follows. ◀

References

1. S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proceedings of Symposium on Discrete Algorithms (SODA, 2012)*, pages 1228–1241, 2012.
2. Nikhil Bansal and Ho-Leung Chan. Weighted flow time does not admit o(1)-competitive algorithms. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, 2009)*, pages 1238–1244, 2009.
3. Nikhil Bansal and Kedar Dhamdhere. Minimizing weighted flow time. *ACM Transactions on Algorithms*, 3(4), 2007.
4. Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 84–93, 2001.
5. Anamitra Roy Choudhury, Syamantak Das, Naveen Garg, and Amit Kumar. Rejecting jobs to minimize load and maximum flow-time. In *Proc. Symposium on Discrete Algorithms*, pages 1114–1133, 2015.
6. Leah Epstein and Rob van Stee. Optimal on-line flow time with resource augmentation. *Discrete Applied Mathematics*, 154(4):611–621, 2006.
7. Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.
8. Giorgio Lucarelli, Benjamin Moseley, Nguyen Kim Thang, Abhinav Srivastav, and Denis Trystram. Online non-preemptive scheduling on unrelated machines with rejections. In *ACM Symposium on Parellelism in Algorithms and Architectures (SPAA, 2018)*, page To appear, 2018.
9. 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.
10. Cynthia A Phillips, Clifford Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.