Online Non-Preemptive Scheduling in a Resource Augmentation Model Based on Duality

Giorgio Lucarelli[*1], Nguyen Kim Thang[†2], Abhinav Srivastav[‡3], and Denis Trystram[§4]

1 **LIG, Université Grenoble-Alpes, Saint-Martin-d'Hères, France**
2 **IBISC, Université d'Evry, Évry, France**
3 **LIG, Université Grenoble-Alpes, Saint-Martin-d'Hères, France; and Verimag, Université Grenoble-Alpes, Saint-Martin-d'Hères, France**
4 **LIG, Université Grenoble-Alpes, Saint-Martin-d'Hères, France**

Abstract

Resource augmentation is a well-established model for analyzing algorithms, particularly in the online setting. It has been successfully used for providing theoretical evidence for several heuristics in scheduling with good performance in practice. According to this model, the algorithm is applied to a more powerful environment than that of the adversary. Several types of resource augmentation for scheduling problems have been proposed up to now, including speed augmentation, machine augmentation and more recently rejection. In this paper, we present a framework that unifies the various types of resource augmentation. Moreover, it allows generalize the notion of resource augmentation for other types of resources. Our framework is based on mathematical programming and it consists of extending the domain of feasible solutions for the algorithm with respect to the domain of the adversary. This, in turn allows the natural concept of duality for mathematical programming to be used as a tool for the analysis of the algorithm's performance. As an illustration of the above ideas, we apply this framework and we propose a primal-dual algorithm for the online scheduling problem of minimizing the total weighted flow time of jobs on unrelated machines when the preemption of jobs is not allowed. This is a well representative problem for which no online algorithm with performance guarantee is known. Specifically, a strong lower bound of $\Omega(\sqrt{n})$ exists even for the offline unweighted version of the problem on a single machine. In this paper, we first show a strong negative result even when speed augmentation is used in the online setting. Then, using the generalized framework for resource augmentation and by combining speed augmentation and rejection, we present an $(1+\epsilon_s)$-speed $O(\frac{1}{\epsilon_s\epsilon_r})$-competitive algorithm if we are allowed to reject jobs whose total weight is an ϵ_r-fraction of the weights of all jobs, for any $\epsilon_s > 0$ and $\epsilon_r \in (0,1)$. Furthermore, we extend the idea for analysis of the above problem and we propose an $(1+\epsilon_s)$-speed ϵ_r-rejection $O\left(\frac{k^{(k+3)/k}}{\epsilon_r^{1/k}\epsilon_s^{(k+2)/k}}\right)$-competitive algorithm for the more general objective of minimizing the weighted ℓ_k-norm of the flow times of jobs.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Sequencing and scheduling

Keywords and phrases Online algorithms, Non-preemptive scheduling, Resource augmentation, Primal-dual

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.63

[*] Giorgio Lucarelli ais partially supported by the ANR projet Moebus n° ANR-13-INFR-0001.
[†] Nguyen Kim Thang is supported by the ANR project OATA n° ANR-15-CE40-0015-01.
[‡] Abhinav Srivastav is partially supported by the LabEx PERSYVAL Lab (ANR-11-LABX-0025-01) funded by the French program "Investissement d'avenir".
[§] Denis Trystram is partially supported by the ANR projet Moebus n° ANR-13-INFR-0001.

1 Introduction

A well-identified issue in algorithms and, in particular, in online computation is the weakness of the worst case paradigm. Summarizing an algorithm by a pathological worst case can underestimate its performance on most inputs. Many practically well-performed algorithms admit a mediocre theoretical guarantee whereas theoretically established algorithms behave poorly even on simple instances in practice. The need of more accurate models is crucial and is considered as an important question in algorithmic community. Several models have been proposed in this direction.

A *first* type of models study online problems assuming nice properties on the inputs. For example, several models, in which arrivals of requests are assumed to follow a given distribution, an unknown distribution, a Markov chain, a random order, etc, have been studied for fundamental online problems such as paging, k-server, matching, Steiner tree. Other models that assume properties on inputs include the access graph model [5], the diffuse adversary model [22], the statistical adversary model [25], the working set model [1, 10]. A *second* type of models consists of giving more power to online algorithms and compare the online algorithm (with additional power) to the offline optimum (without additional power). This class consists of the model with advice [12, 13] and the resource augmentation model [20, 24]. A *third* type of models aim at comparing an online algorithm to some benchmark different from the offline optimum. This class includes the comparative analysis [22], the bijective analysis [3], etc. Each model has successfully explained the performance of algorithms in certain contexts but it has limits against other classes of problems. The lack of appropriate tools is a primary obstacle for the advance of most of the above models.

In this paper, we are interested in studying the resource augmentation model that compares online algorithms to a weaker adversary. Kalyanasundaram and Pruhs [20] proposed a *speed augmentation* model, where an online algorithm is compared against an adversary with slower processing speed. Phillips et al. [24] proposed the *machine augmentation* model in which the algorithm has more machines than the adversary. Recently, Choudhury et al. [8] introduced the *rejection model* where an online algorithm is allowed to discard a small fraction of jobs. The power of these models lies in the fact that many natural scheduling algorithms can be analyzed with respect to them, as well as, they have successfully provided theoretical evidence for heuristics in scheduling with good performance in practice. Although the models give more power to online algorithms, the connection especially between the latter and the two formers is unclear and the disconnection is emphasized by the fact that some algorithms have good performance in a model but have moderate behavior in others (for example, the problem of minimizing maximum flow-time [8]).

1.1 Generalized Resource Augmentation and Approach

In this paper, we introduce a generalized resource augmentation model that unifies all the previous ones. We also consider an approach based on duality for the systematic study of algorithms in this new model. To see that the duality is particularly appropriate, we first explain the model and the approach intuitively.

The weak duality in mathematical programming can be interpreted as a game between an algorithm and an adversary (the primal program against the dual one). The game is $L(x, \lambda)$, the standard Lagrangian function completely defined for a given problem, in which x and λ are primal and dual variables, respectively. The primal and dual variables are controlled and correspond to the strategies of the adversary and the algorithm, respectively. The goal of the algorithm is to choose a strategy λ among its feasible sets so as to minimize $L(x, \lambda)$ for

whatever feasible strategy x of the adversary. The algorithm is c-competitive if it can choose dual variables λ^* in such a way that whatever the strategy (choice on x) of the adversary, the value $\min_x L(x, \lambda^*)$ is always within a desirable factor c of the objective due to the algorithm.

The resource augmentation models [8, 20] consist in giving more power to the algorithm. This idea could be perfectly interpreted as a game between an algorithm and an adversary in which additional power for the algorithm is reflected by better choices over its feasible strategy set.

Concretely, let us illustrate this idea for the speed augmentation and the rejection models. In several scheduling problems, a constraint originally states that the speed of a given machine is at most one. In the speed augmentation model, this constraint is relaxed such that the algorithm executes jobs at higher speed than that of the adversary. On other hand, the relaxation is of a different nature in the rejection model. Specifically, there are usually constraints ensuring that all jobs should be completed. In the rejection model, the algorithm is allowed to systematically reject a fraction of constraints whereas adversary should satisfy all of them. In both models, the algorithm optimizes the objective over a feasible domain whereas the adversary optimizes the same objective over a sub-domain with respect to the algorithm. This naturally leads to a more general model of resource augmentation.

▶ **Definition 1** (Generalized Resource Augmentation)**.** Consider an optimization problem that can be formalized by a mathematical program. Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q}$ be a subset of $\mathcal{P}$. In *generalized resource augmentation*, the performance of an algorithm is measured by the worst ratio between its objective over $\mathcal{P}$ and that of a solution which is optimized over $\mathcal{Q}$.

Based on the above definition, the polytope of the adversary in speed augmentation model is a strict subset of the algorithm's polytope since the speed constraint for the adversary is tighter. In the rejection model, the polytope of the adversary is also a strict subset of the algorithm's one since it contains more constraints. In addition, the generalized model allows us to introduce different kind of relaxations to the set of feasible solutions – each corresponding to different type of augmentations.

Together with the generalized model, we consider the following duality-based approach for the systematic design and analysis of algorithms. Let $\mathcal{P}$ and $\mathcal{Q}$ be the sets of feasible solutions for the algorithm and the adversary, respectively. By resource augmentation, $\mathcal{Q} \subset \mathcal{P}$. In order to study the performance of an algorithm, we consider the dual of the mathematical program consisting of the objective function optimized over $\mathcal{Q}$. By weak duality, the dual is a lower bound for any solution. Then, we bound the algorithm's cost by that of this dual. We exploit the resource augmentation properties (relation between $\mathcal{P}$ and $\mathcal{Q}$) to derive effective bounds. Intuitively, one needs to take the advantage from resource augmentation so as to raise the dual as much as possible — an impossible procedure without resource augmentation. As it has been shown in previous works and as we will see below, the duality approach is particularly appropriate to study problems with resource augmentation.

1.2 Our Contributions

We illustrate the applicability of the generalized model and the duality-based approach through a scheduling problem, in which jobs arrive online and they have to be scheduled non-preemptively on a set of unrelated machines. The objective is to minimize the average weighted time a job remains in the system (average weighted flow-time), where the weights represent the importance of the jobs. This is a well representative hard problem since no online algorithm with performance guarantee is known. Specifically, a strong lower bound of

$\Omega(\sqrt{n})$ exists even for the offline unweighted version of the problem on a single machine [21], where n is the number of jobs. For the online setting, any algorithm without resource augmentation has at least $\Omega(n)$ competitive ratio, even for single machine (as mentioned in [7]). Moreover, in contrast to the preemptive case, our first result (Lemma 2) shows that no deterministic algorithm has bounded competitive ratio when preemptions are not allowed even if we consider a single machine which has arbitrary large speed augmentation. However, the non-preemptive scheduling is a natural setting and it is important to have algorithms with theoretical explanation on their performance or a mean to classify algorithms.

In this paper, we present a competitive algorithm in a model which combines speed augmentation and the rejection model. Specifically, for arbitrary $0 < \epsilon_r < 1$ and $\epsilon_s > 0$, there exists a $O(1/(\epsilon_r \cdot \epsilon_s))$-competitive algorithm that uses machines with speed $(1+\epsilon_s)$ and rejects jobs with at most ϵ_r-fraction of the total weight of all jobs. The design and analysis of the algorithm follow the duality approach. At the release time of any job j, the algorithm defines the dual variables associated to the job and assigns the job to some machine based on this definition. The value of the dual variables associated to a job j are selected in order to satisfy two key properties: (i) comprise the marginal increase of the total weighted flow-time due to the arrival of the job — the property that has been observed [2, 26] and has become more and more popular in dual-fitting for online scheduling; and (ii) capture the information for a future decision of the algorithm whether job j will be completed or rejected — a *novel* point in the construction of dual variables to exploit the power of rejection. Informally, to fulfill the second property, we introduce *prediction* terms to dual variables that at some point in the future will indicate whether the corresponding job would be rejected. Moreover, these terms are chosen so as to stabilize the schedule such that the properties of the assignment policy are always preserved (even with job rejections in the future). This allows us to maintain a non-migratory schedule.

Our algorithm dispatches jobs immediately at their release time — a desired property in scheduling. Besides, the algorithm processes jobs in the highest density first manner and interrupts a job only if it is rejected. In other words, no completed job has been interrupted during its execution. The algorithm is relatively simple, particularly for a single machine setting as there is no assignment policy. Therefore, the analysis of the algorithm in the generalized resource augmentation could be considered as a first step toward the theoretical explanation for the well-known observation that simple scheduling algorithms usually behave well and are widely used in practice.

Finally, we extend the above ideas to the more general objective of minimizing the weighted ℓ_k-norm of flow-time of jobs on unrelated machines. The ℓ_k-norm captures the notion of fairness between jobs since it removes the extreme outliers and hence it is more appropriate to balance the difference among the flow-times of individual jobs than the average function, which corresponds to the ℓ_1-norm (see for example [23]). For the ℓ_k-norm objective, we propose a primal-dual algorithm which is $(1+\epsilon_s)$-speed $O\left(\frac{k^{(k+3)/k}}{\epsilon_r^{1/k}\epsilon_s^{(k+2)/k}}\right)$-competitive and it rejects jobs of total weight at most ϵ_r-fraction of the total weight of all jobs. The analysis for this problem is more technical and it is given in the Appendix.

1.3 Related Work

Duality based techniques have been extensively developed in approximation algorithms [27] and in online algorithms [6]. Specifically, Buchbinder and Naor [6] gave a general framework for online covering/packing LPs that applies to several fundamental problems in online computation. However, this framework encounters different issues to design competitive

algorithms for online scheduling problems. Recently, Anand et al. [2] have proposed the use of dual-fitting techniques to study scheduling problems in the speed augmentation model. After this seminal paper, the duality approaches in online scheduling have been extended to a variety of problems, and has rapidly become standard techniques. The duality approaches have also led to the development of newer techniques for analyzing algorithm (see for example [2, 11, 15, 16, 18, 17, 26]). Informally, in the approach proposed in [2], the key step relies on the construction of a dual feasible solution in such a way that its dual objective is up to some bounded factor from that of the algorithm. In [2], the dual variables are carefully designed in order to encode the power of speed augmentation. Later on, Nguyen [26] explicitly formalized the comparison through the mean of Lagrangian functions between the algorithm and the adversary, with a tighter feasible domain due to speed augmentation. That point of view makes the framework in [2] effective to study non-convex formulations.

For the online non-preemptive scheduling problem of minimizing total weighted flow-time, no competitive algorithm for unrelated machines even with resource augmentation is known; that is in contrast to the preemptive version which has been well studied [2, 11, 16, 18, 17, 26]. For identical machines, Phillips et al. [24] gave a constant competitive algorithm that uses $m \log P$ machines (recall that the adversary uses m machines), where P is the ratio of the largest to the smallest processing time. Moreover, an $O(\log n)$-machine $O(1)$-speed algorithm that returns the optimal schedule has been presented in [24] for the unweighted flow-time objective. Epstein and van Stee [14] 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. For the offline non-preemptive single machine setting, Bansal et al. [4] gave a 12-speed $(2+\epsilon)$-approximation polynomial time algorithm. Recently, Im et al. [19] gave a $(1+\epsilon)$-speed $(1+\epsilon)$-approximation quasi-polynomial time algorithm for the setting of identical machines.

For the online non-preemptive scheduling problem of minimizing the weighted ℓ_k-norm of flow-time, to the best of our knowledge, no competitive algorithm is known. However, the problem in the preemptive setting has been widely studied. With speed augmentation, Anand et al. [2] gave a $(1+\epsilon)$-speed, $O(k/\epsilon^{2+1/k})$-competitive algorithm but the algorithm needs to know the speed $(1+\epsilon)$ in advance. Later on, Nguyen [26] derived an improved $(1+\epsilon)$-speed, $k/\epsilon^{1+1/k}$-competitive algorithm which does not need information on ϵ a priori. Recently, Choudhury et al. [9] have considered the (preemptive) problem in the restricted assignment setting in the rejection model. They have presented a $1/\epsilon^{O(1)}$-competitive algorithm that rejects at most ϵ fraction of the total job weight.

2 Problem Definition and Notation

We are given a set $\mathcal{M}$ of m unrelated machines. The jobs arrive *online*, that is we learn about the existence and the characteristics of a job only after its release. Let $\mathcal{J}$ denote the set of all jobs of our instance, which is not known a priori. Each job $j \in \mathcal{J}$ is characterized by its *release time* r_j, its *weight* w_j and if job j is executed on machine $i \in \mathcal{M}$ then it has a *processing time* p_{ij}. We study the *non-preemptive* setting, meaning that a job is considered to be completed only if it is fully processed in one machine without interruption during its execution. This definition allows the interruption of jobs. However, if the execution of a job is interrupted then it has to be processed entirely later on in order to be considered as completed. In this paper, we consider a stronger non-preemptive model according to which we are only allowed to interrupt a job if we reject it, i.e., we do not permit restarts. Moreover, each job has to be dispatched to one machine at its arrival and migration is not

allowed. Given a schedule $\mathcal{S}$, we denote by C_j the *completion time* of the job j. Then, its *flow-time* is defined as $F_j = C_j - r_j$, that is the total time that j remains in the system. Our objective is to create a non-preemptive schedule that minimizes the total weighted flow-times of all jobs, i.e., $\sum_{j\in\mathcal{J}} w_j F_j$. A more general objective that implies fairness between jobs is the minimization of the weighted ℓ_k-norm of the flow-time of all jobs, i.e., $(\sum_{j\in\mathcal{J}} w_j F_j^k)^{1/k}$, where $k \geq 1$.

Let $\delta_{ij} = \frac{w_j}{p_{ij}}$ be the *density* of a job $j \in \mathcal{J}$ on machine $i \in \mathcal{M}$. Moreover, let $q_{ij}(t)$ be the remaining processing time at time t of a job $j \in \mathcal{J}$ which is dispatched at machine $i \in \mathcal{M}$. A job $j \in \mathcal{J}$ is called *pending* at time t, if it is already released at t but not yet completed, i.e., $r_j \leq t < C_j$. Finally, let $P = \max_{j,j'\in\mathcal{J}}\{p_j/p_{j'}\}$ and $W = \max_{j,j'\in\mathcal{J}}\{w_j/w_{j'}\}$.

3 Scheduling to Minimize Total Weighted Flow-time

In this section, we describe our primal-dual method for the online non-preemptive scheduling problem of minimizing the total weighted flow-time on unrelated machines. This problem admits no competitive algorithm even with speed augmentation as shown by the following lemma.

▶ **Lemma 2.** *For any speed augmentation $s \leq P^{1/10}$ or $s < W^{1/6}$, every deterministic algorithm has competitive ratio at least $\Omega(P^{1/10})$ or $\Omega(W^{1/6})$, respectively, even for the single machine problem.*

Proof. Let $s > 1$ be the speed of the machine; without loss of generality we assume that the machine speed for the adversary is 1. Let $R > s^2$ be an arbitrary (large) constant and fix an algorithm.

We consider the following instance. At time 1, a *long* job of processing time $2sR^3$ and weight 1 is released. After that, the phase 1 starts: at any time aR^3, starting with $a = 1$, a *short* job of processing time 1 and weight R is released. If the algorithm processes the long job during the whole interval $[aR^3, (a+1)R^3]$, then the adversary stops releasing jobs and the instance halts. Otherwise, the adversary will release a new short job at time $(a+1)R^3$ and so on, until $a = 2s - 1$. Then, the phase 2 begins immediately after phase 1: at any time aR^3 for $a \geq 2s$, a *small* job of processing time 1 and weight R^2 is released. Similarly, if the algorithm keeps processes the long job during the whole interval $[aR^3, (a+1)R^3]$, the instance halts. Otherwise, the adversary will release a new small job at time $(a+1)R^3$, until $a = 2sR^2$.

In the instance, we have at most $2s$ short jobs and $2sR^2$ small jobs. Observe that by using speed s, the algorithm cannot complete the long job between two consecutive release times of short or small jobs. We analyze the performance of the algorithm by considering different cases.

Case 1: the instance halts during phase 1. In this case, there is a $a \in \{1, 2, \ldots, 2s-1\}$ for which the algorithm keeps processing the long job during the whole interval $[aR^3, (a+1)R^3]$ and hence the short job released at aR^3 is not processed during that time interval. Thus, the weighted flow-time of the short job is at least $R \cdot R^3$. However, the adversary can execute immediately all short jobs at their release times and process the long job in the end. The total weighted flow-time of all short jobs is at most $2sR$. The long job would be started no later than the time where phase 1 terminates, which is $(2s-1)R^3 + 1$. So the weighted flow-time of the long job is at most $4sR^3$. Therefore, the competitive ratio is at least $\Omega(R/s)$.

Case 2: the instance halts during phase 2. In this case, there is a $a \in \{2s, 2s+1, \ldots, 2sR^2\}$ for which the algorithm keeps processing the long job during the whole interval $[aR^3, (a+1)R^3]$ and hence the small job released at aR^3 is not processed during that time interval. We proceed similarly as in the previous case. The weighted flow-time of this small job is at least $R^2 \cdot R^3$. Nevertheless, the adversary can process the long job during $[1, 2sR^3 + 1]$, execute small jobs at their release time (except the first one which starts 1 unit of time after its release time) and execute all short jobs during the interval $[2sR^3 + 2, 5sR^3]$ whenever a small job is not executed. This is a feasible schedule since the number of short jobs is $(2s-1) < 3R^3 - 5$ (note that there are 2 small jobs released during $[2sR^3+2, 5sR^3]$). By this strategy, the weighted flow-time of the long job is $2sR^3 + 1$. The total weighted flow-time of small jobs is at most $2sR^2 \cdot R^2$. The total weighted flow-time of short jobs is at most $2s \cdot R \cdot 5sR^3$. Hence, the cost of the adversary is at most $14s^2R^4$ and the competitive ratio is at least $\Omega(R/s^2)$.

Case 3: the instance halts at the end of phase 2. The algorithm executes the long job after the end of phase 2 and hence this job is completed at later than $2sR^5$; so its weighted flow-time is at least $2sR^5$. The adversary can apply the same strategy as in Case 2 with total cost $14s^2R^4$. Therefore, the competitive ratio is at least $\Omega(R/s)$.

In summary, the competitive ratio is at least $\Omega(R/s^2)$. Recall that P and W are the largest ratio between processing times and that between weights, respectively. In this instance, $P = 2sR^3$ and $W = R^2$ respectively. By a simple estimation (setting $R = s^3$), for any speed $s \leq P^{1/10}$ the competitive ratio is at least $\Omega(P^{1/10})$; and for $s \leq W^{1/6}$, the competitive ratio is at least $\Omega(W^{1/6})$. ◀

In the following, we study the problem in the resource augmentation model with speed augmentation and rejection.

3.1 Linear Programming Formulation

For each machine $i \in \mathcal{M}$, job $j \in \mathcal{J}$ and time $t \geq r_j$, we introduce a binary variable $x_{ij}(t)$ which indicates if j is processed on i at time t. We consider the following linear programming formulation. Note that the objective value of this linear program is at most twice that of the optimal preemptive schedule.

$$\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \delta_{ij} \left(t - r_j + p_{ij}\right) x_{ij}(t) dt$$

$$\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \quad \forall j \in \mathcal{J} \tag{1}$$

$$\sum_{j \in \mathcal{J}} x_{ij}(t) \leq 1 \quad \forall i \in \mathcal{M}, t \tag{2}$$

$$x_{ij}(t) \in \{0, 1\} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j$$

After relaxing the integrality constraints, we get the following dual program.

$$\max \sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_0^{\infty} \gamma_i(t) dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \delta_{ij} \left(t - r_j + p_{ij}\right) \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j \tag{3}$$

We will interpret the resource augmentation models in the above primal and dual programs as follows. In the speed augmentation model, we assume that all machines in the schedule of our algorithm run with speed 1, while in adversary's schedule they run at a speed $a < 1$. This can be interpreted in the primal linear program by modifying the constraint (2) to be $\sum_{j\in\mathcal{J}} x_{ij}(t) \leq a$. Intuitively, each machine in the adversary's schedule can execute jobs with speed at most a at each time t. The above modification in the primal program reflects to the objective of the dual program which becomes $\sum_{j\in\mathcal{J}} \lambda_j - a\sum_{i\in\mathcal{M}} \int_0^\infty \gamma_i(t)dt$. In the rejection model, we assume that the algorithm is allowed to reject some jobs. This can be interpreted in the primal linear program by summing up only on the set of the non rejected jobs, i.e., the algorithm does not have to satisfy the constraint (1) for rejected jobs. Hence the objective becomes $\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}\setminus\mathcal{R}} \int_{r_j}^\infty \delta_{ij}\,(t - r_j + p_{ij})\,dt$. Concluding, our algorithm rejects a set $\mathcal{R}$ of jobs, uses machines with speed $1/a$ times faster than that of the adversary and, by using weak duality, has a competitive ratio at most

$$\frac{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}\setminus\mathcal{R}} \int_{r_j}^\infty \delta_{ij}(t - r_j + p_{ij})dt}{\sum_{j\in\mathcal{J}} \lambda_j - a\sum_{i\in\mathcal{M}} \int_0^\infty \gamma_i(t)dt}.$$

3.2 Algorithm and Dual Variables

We describe next the scheduling, the rejection and the dispatching policies of our algorithm which we denote by $\mathcal{A}$. In parallel, we give the intuition about the definition of the dual variables in a primal-dual way. Let $\epsilon_s > 0$ and $0 < \epsilon_r < 1$ be constants arbitrarily small. Intuitively, ϵ_s and ϵ_r stand for the speed augmentation and the rejection fraction of our algorithm, respectively. In what follows, we assume that in the schedule created by $\mathcal{A}$ all machines run with speed 1, while in the adversary's schedule they run by speed $\frac{1}{1+\epsilon_s}$.

Each job is immediately dispatched to a machine upon its arrival. We denote by $Q_i(t)$ the set of pending jobs at time t dispatched to machine $i \in \mathcal{M}$, i.e., the set of jobs dispatched to i that have been released but not yet completed and have not been rejected at t. Our *scheduling policy* for each machine $i \in \mathcal{M}$ is the following: at each time t when the machine i becomes idle or has just completed or interrupted some job, we start executing on i the job $j \in Q_i(t)$ such that j has the largest density in $Q_i(t)$, i.e., $j = \text{argmax}_{j'\in Q_i(t)}\{\delta_{ij'}\}$. In case of ties, we select the job that arrived earliest.

When a machine $i \in \mathcal{M}$ starts executing a job $k \in \mathcal{J}$, we introduce a counter v_k (associated to job k) which is initialized to zero. Each time when a job $j \in \mathcal{J}$ with $\delta_{ij} > \delta_{ik}$ is released during the execution of k and j is dispatched to i, we increase v_k by w_j. Then, the *rejection policy* is the following: we interrupt the execution of the job k and we reject it the first time where $v_k > \frac{w_k}{\epsilon_r}$.

Let Δ_{ij} be the increase in the total weighted flow-time occurred in the schedule of our algorithm if we assign a new job $j \in \mathcal{J}$ to machine i, following the above scheduling and rejection policies. Assuming that the job $k \in \mathcal{J}$ is executed on i at time r_j, we have that

$$\Delta_{ij} = \begin{cases} w_j\left(q_{ik}(r_j) + \sum\limits_{\substack{\ell\in Q_i(r_j)\setminus\{k\}:\\ \delta_{i\ell}\geq\delta_{ij}}} p_{i\ell}\right) + p_{ij}\sum\limits_{\substack{\ell\in Q_i(r_j)\setminus\{k\}:\\ \delta_{i\ell}<\delta_{ij}}} w_\ell & \text{if } v_k + w_j \leq \frac{w_k}{\epsilon_r}, \\ w_j\sum\limits_{\substack{\ell\in Q_i(r_j):\\ \delta_{i\ell}\geq\delta_{ij}}} p_{i\ell} + \left(p_{ij}\sum\limits_{\substack{\ell\in Q_i(r_j):\\ \delta_{i\ell}<\delta_{ij}}} w_\ell - q_{ik}(r_j)\sum\limits_{\substack{\ell\in Q_i(r_j)\cup\{k\}:\\ \ell\neq j}} w_\ell\right) & \text{otherwise.} \end{cases}$$

where, in both cases, the first term corresponds to the weighted flow-time of the job j if it is dispatched to i and the second term corresponds to the change of the weighted flow-time for

the jobs in $Q_i(r_j)$. Note that, the second case corresponds to the rejection of k and hence we remove the term $w_j q_{ik}(r_j)$ in the weighted flow-time of j, while the flow-time of each pending job is reduced by $q_{ik}(r_j)$.

In the definition of the dual variables, we aim to charge to job j the increase Δ_{ij} in the total weighted flow-time occurred by the dispatching of j in machine i, except from the quantity $w_j q_{ik}(r_j)$ which will be charged to job k, if $\delta_{ij} > \delta_{ik}$. However, we will use the dual variables (in the primal-dual sense) to guide the assignment policy. Hence the charges have to be distributed in a consistent manner to the assignment decisions of jobs to machines in the past. So in order to do the charging, we introduce a *prediction term*: at the arrival of each job j we charge to it an additional quantity of $\frac{w_j}{\epsilon_r} p_{ij}$. By doing this, the consistency is maintained by the rejection policy: if the charge from future jobs exceeds the prediction term of some job then the latter will be rejected.

Based on the above, we define

$$\lambda_{ij} = \begin{cases} \frac{w_j}{\epsilon_r} p_{ij} + w_j \sum\limits_{\ell \in Q_i(r_j):\delta_{i\ell} \geq \delta_{ij}} p_{i\ell} + p_{ij} \sum\limits_{\ell \in Q_i(r_j)\setminus\{k\}:\delta_{i\ell} < \delta_{ij}} w_\ell & \text{if } \delta_{ij} > \delta_{ik} \\ \frac{w_j}{\epsilon_r} p_{ij} + w_j \left(q_{ik}(r_j) + \sum\limits_{\ell \in Q_i(r_j)\setminus\{k\}:\delta_{i\ell} \geq \delta_{ij}} p_{i\ell} \right) + p_{ij} \sum\limits_{\ell \in Q_i(r_j):\delta_{i\ell} < \delta_{ij}} w_\ell & \text{otherwise} \end{cases}$$

which represents the total charge for job j if it is dispatched to machine i. Note that the only difference in the two cases of the definition of λ_{ij} is that we charge the quantity $w_j q_{ik}(r_j)$ to j only if $\delta_{ij} \leq \delta_{ik}$. Moreover, we do not consider the negative quantity that appears in the second case of Δ_{ij}. Intuitively, we do not decrease our estimation for the completion times of pending jobs when a job is rejected. The *dispatching policy* is the following: dispatch j to the machine $i^* = \text{argmin}_{i \in \mathcal{M}}\{\lambda_{ij}\}$. Intuitively, a part of Δ_{ij} may be charged to job k, and more specifically to the prediction part of λ_{ik}. However, we do not allow to exceed this prediction by applying rejection. In other words, the rejection policy can be re-stated informally as: we reject k just before we exceed the prediction charging part in λ_{ik}.

In order to keep track of the negative terms in Δ_{ij}, for each job $j \in \mathcal{J}$ we denote by D_j the set of jobs that are rejected by the algorithm after the release time of j and before its completion or rejection (including j in case it is rejected), that is the jobs that cause a decrease to the flow time of j. Moreover, we denote by j_k the job released at the moment we reject a job $k \in \mathcal{R}$. Then, we say that a job $j \in \mathcal{J}$ which is dispatched to machine $i \in \mathcal{M}$ is *definitively finished* $\sum_{k \in D_j} q_{ik}(r_{j_k})$ time after its completion or rejection. Let $U_i(t)$ be the set of jobs that are dispatched to machine $i \in \mathcal{M}$, they are already completed or rejected at a time before t, but they are not yet definitively finished at t.

It remains to formally define the dual variables. At the arrival of a job $j \in \mathcal{J}$, we set $\lambda_j = \frac{\epsilon_r}{1+\epsilon_r} \min_{i \in \mathcal{M}}\{\lambda_{ij}\}$ and we never change λ_j again. Let $W_i(t)$ be the total weight of jobs dispatched to machine $i \in \mathcal{M}$ in the schedule of $\mathcal{A}$, and either they are pending at t or they are not yet definitively finished at t, i.e., $W_i(t) = \sum_{\ell \in Q_i(t) \cup U_i(t)} w_\ell$. Then, we define $\gamma_i(t) = \frac{\epsilon_r}{1+\epsilon_r} W_i(t)$. Note that $\gamma_i(t)$ is updated during the execution of $\mathcal{A}$. Specifically, given any fixed time t, $\gamma_i(t)$ may increase if a new job j' arrives at any time $r_{j'} \in [r_j, t)$. However, $\gamma_i(t)$ does never decrease in the case of rejection since the jobs remain in $U_i(t)$ for a sufficient time after their completion or rejection.

3.3 Analysis

We first prove the following lemma which guarantees the feasibility of the dual constraint using the above definition of the dual variables.

▶ **Lemma 3.** *For every machine $i \in \mathcal{M}$, job $j \in \mathcal{J}$ and time $t \geq r_j$, the dual constraint is feasible, that is*

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) - \delta_{ij}\,(t - r_j + p_{ij}) \leq 0\,.$$

Proof. Fix a machine i. We have observed above that, for any fixed time $t \geq r_j$, as long as new jobs arrive, the value of $\gamma_i(t)$ may only increase. Then, it is sufficient to prove the dual constraints for the job j using the values of $\gamma_i(t)$, $Q_i(t)$, $U_i(t)$ and $W_i(t)$ as these are defined at time r_j. Let k be the job executed in machine i at r_j. We have the following cases.

Case 1: $\delta_{ij} > \delta_{ik}$

In this case we may have rejected the job k at r_j. By the definitions of λ_j and λ_{ij}, we have

$$\begin{aligned}\frac{\lambda_j}{p_{ij}} &\leq \frac{\epsilon_r}{(1+\epsilon_r)}\frac{\lambda_{ij}}{p_{ij}} = \frac{\epsilon_r}{1+\epsilon_r}\left(\frac{w_j}{\epsilon_r} + \delta_{ij}\sum_{\ell\in Q_i(r_j):\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + \sum_{\ell\in Q_i(r_j)\setminus\{k\}:\delta_{i\ell}<\delta_{ij}} w_\ell\right)\\ &= \frac{\epsilon_r}{1+\epsilon_r}\left(\frac{w_j}{\epsilon_r} + \delta_{ij}\sum_{\ell\in Q_i(r_j)\setminus\{j\}:\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + w_j + \sum_{\ell\in Q_i(r_j)\setminus\{k\}:\delta_{i\ell}<\delta_{ij}} w_\ell\right)\\ &= w_j + \frac{\epsilon_r}{1+\epsilon_r}\left(\delta_{ij}\sum_{\ell\in Q_i(r_j)\setminus\{j\}:\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + \sum_{\ell\in Q_i(r_j)\setminus\{k\}:\delta_{i\ell}<\delta_{ij}} w_\ell\right)\end{aligned}$$

By the definition of $\gamma_i(t)$ we get

$$\begin{aligned}\gamma_i(t) + \delta_{ij}(t - r_j + p_{ij}) &= \frac{\epsilon_r}{1+\epsilon_r}\sum_{\ell\in Q_i(t)\cup U_i(t)} w_\ell + \delta_{ij}(t - r_j) + w_j\\ &\geq \frac{\epsilon_r}{1+\epsilon_r}\left(\sum_{\ell\in Q_i(t)\cup U_i(t)} w_\ell + \delta_{ij}(t - r_j)\right) + w_j\end{aligned}$$

Thus, it remains to show that

$$\delta_{ij}\cdot\sum_{\substack{\ell\in Q_i(r_j)\setminus\{j\}:\\ \delta_{i\ell}\geq\delta_{ij}}} p_{i\ell} \;+\; \sum_{\substack{\ell\in Q_i(r_j)\setminus\{k\}:\\ \delta_{i\ell}<\delta_{ij}}} w_\ell \leq \sum_{\ell\in Q_i(t)\cup U_i(t)} w_\ell \;+\delta_{ij}(t - r_j) \tag{4}$$

Let $\tilde{C}_j = r_j + \sum_{\ell\in Q_i(r_j):\delta_{i\ell}\geq\delta_{ij}} p_{i\ell}$ (if k is rejected) or $\tilde{C}_j = r_j + q_{ik}(r_j) + \sum_{\ell\in Q_i(r_j):\delta_{i\ell}\geq\delta_{ij}} p_{i\ell}$ (otherwise) be the estimated completion time of j at time r_j if it is dispatched to machine i.

Case 1.1: $t \leq \tilde{C}_j$. By the definition of $U_i(t)$, all jobs in $Q_i(r_j)$ with $\delta_{i\ell} < \delta_{ij}$ still exist in $Q_i(t) \cup U_i(t)$. Moreover, for every job $\ell \in Q_i(r_j) \setminus (Q_i(t) \cup U_i(t) \cup \{k\})$ it holds that $\delta_{i\ell} \geq \delta_{ij}$, since ℓ is processed before j by the algorithm. Then, by splitting the first term of

the left-hand side of (4) we get

$$\begin{aligned}
&\delta_{ij} \cdot \sum_{\ell \in Q_i(r_j)\setminus\{j\}:\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + \sum_{\ell \in Q_i(r_j)\setminus\{k\}:\delta_{i\ell}<\delta_{ij}} w_\ell \\
&= \delta_{ij} \sum_{\ell \in Q_i(r_j)\setminus(Q_i(t)\cup U_i(t)\cup\{k\})} p_{i\ell} + \delta_{ij} \sum_{\substack{\ell \in (Q_i(r_j)\cap(Q_i(t)\cup U_i(t)))\setminus\{j\}:\\ \delta_{i\ell}\geq\delta_{ij}}} p_{i\ell} \\
&\qquad + \sum_{\substack{\ell \in (Q_i(r_j)\cap(Q_i(t)\cup U_i(t)))\setminus\{k\}:\\ \delta_{i\ell}<\delta_{ij}}} w_\ell \\
&\leq \delta_{ij} \sum_{\ell \in Q_i(r_j)\setminus(Q_i(t)\cup U_i(t)\cup\{k\})} p_{i\ell} + \sum_{\substack{\ell \in (Q_i(t)\cup U_i(t))\setminus\{j\}:\\ \delta_{i\ell}\geq\delta_{ij}}} w_\ell + \sum_{\substack{\ell \in (Q_i(t)\cup U_i(t))\setminus\{k\}:\\ \delta_{i\ell}<\delta_{ij}}} w_\ell \\
&\leq \delta_{ij}(t - r_j) + \sum_{\ell \in Q_i(t)\cup U_i(t)} w_\ell
\end{aligned}$$

where the first inequality is due to $\delta_{ij}p_{i\ell} \leq w_\ell$ for each $\ell \in Q_i(t) \cup U_i(t)$ with $\delta_{i\ell} \geq \delta_{ij}$, while the latter one holds since the set of jobs $Q_i(r_j) \setminus (Q_i(t) \cup U_i(t) \cup \{k\})$ corresponds to the set of pending jobs at r_j that start their execution after r_j and are definitively finished before t.

Case 1.2: $t > \tilde{C}_j$. By splitting the second term of the left-hand side of (4) we get

$$\begin{aligned}
&\delta_{ij} \cdot \sum_{\ell \in Q_i(r_j)\setminus\{j\}:\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + \sum_{\ell \in Q_i(r_j)\setminus\{k\}:\delta_{i\ell}<\delta_{ij}} w_\ell \\
&= \delta_{ij} \sum_{\substack{\ell \in Q_i(r_j)\setminus\{j\}:\\ \delta_{i\ell}\geq\delta_{ij}}} p_{i\ell} + \sum_{\substack{\ell \in Q_i(r_j)\setminus(Q_i(t)\cup U_i(t)\cup\{k\}):\\ \delta_{i\ell}<\delta_{ij}}} w_\ell + \sum_{\substack{\ell \in Q_i(r_j)\cap(Q_i(t)\cup U_i(t)):\\ \delta_{i\ell}<\delta_{ij}}} w_\ell \\
&\leq \delta_{ij}(\tilde{C}_j - r_j) + \delta_{ij} \sum_{\ell \in Q_i(r_j)\setminus(Q_i(t)\cup U_i(t)\cup\{k\}):\delta_{i\ell}<\delta_{ij}} p_{i\ell} + \sum_{\ell \in Q_i(t)\cup U_i(t)} w_\ell \\
&\leq \delta_{ij}(\tilde{C}_j - r_j) + \delta_{ij}(t - \tilde{C}_j) + \sum_{\ell \in Q_i(t)\cup U_i(t)} w_\ell
\end{aligned}$$

where the first inequality follows by the definition of $\tilde{C}_j$ and since $w_\ell < \delta_{ij}p_{i\ell}$ for each $\ell \in Q_i(r_j)$ with $\delta_{i\ell} < \delta_{ij}$, while the second inequality follows since the set of jobs in $Q_i(r_j) \setminus (Q_i(t) \cup U_i(t) \cup \{k\})$ with $\delta_{i\ell} < \delta_{ij}$ corresponds to the pending jobs at r_j which at time r_j have been scheduled to be executed during the interval $[\tilde{C}_j, t)$.

Case 2: $\delta_{ij} \leq \delta_{ik}$

In this case the job k is not rejected at the arrival of job j. By using the same arguments as in Case 1, we have

$$\frac{\lambda_j}{p_{ij}} \leq w_j + \frac{\epsilon_r}{1+\epsilon_r}\left(\delta_{ij}q_{ik}(r_j) + \delta_{ij} \sum_{\ell \in Q_i(r_j)\setminus\{k,j\}:\delta_{i\ell}\geq\delta_{ij}} p_{i\ell} + \sum_{\ell \in Q_i(r_j):\delta_{i\ell}<\delta_{ij}} w_\ell\right)$$

Let $\tilde{C}_k = r_j + q_{ik}(r_j)$ be the estimated completion time of k at time r_j. We consider different scenarios.

Case 2.1: $t \leq \tilde{C}_k$. In this case, it holds that $w_k \geq \delta_{ij} p_k \geq \delta_{ij} q_{ik}(r_j)$. Then,

$$\gamma_i(t) + \delta_{ij}\left(t - r_j + p_{ij}\right) \geq \frac{\epsilon_r}{1+\epsilon_r} \sum_{\ell \in Q_i(t) \cup U_i(t)} w_\ell + w_j \geq \frac{\epsilon_r}{1+\epsilon_r} \sum_{\ell \in Q_i(r_j)} w_\ell + w_j$$
$$\geq \frac{\epsilon_r}{1+\epsilon_r} \left(\sum_{\ell \in Q_i(r_j) \setminus \{k\}} w_\ell + w_k \right) + w_j \geq \frac{\epsilon_r}{1+\epsilon_r} \left(\sum_{\ell \in Q_i(r_j) \setminus \{k\}} w_\ell + \delta_{ij} q_{ik}(r_j) \right) + w_j$$

Hence, it remains to show

$$\delta_{ij} \sum_{\ell \in Q_i(r_j) \setminus \{k,j\} : \delta_{i\ell} \geq \delta_{ij}} p_{i\ell} + \sum_{\ell \in Q_i(r_j) : \delta_{i\ell} < \delta_{ij}} w_\ell - \sum_{\ell \in Q_i(r_j) \setminus \{k\}} w_\ell \leq 0$$

which directly holds as $\delta_{ij} p_{i\ell} \leq w_\ell$ for any job $j \in Q_i(r_j)$ with $\delta_{i\ell} \geq \delta_{ij}$.

Case 2.2: $t > \tilde{C}_k$. By the definition of $\gamma_i(t)$ we get

$$\gamma_i(t) + \delta_{ij}\left(t - r_j + p_{ij}\right) \geq \frac{\epsilon_r}{1+\epsilon_r} \left(\sum_{\ell \in Q_i(t) \cup U_i(t)} w_\ell + \delta_{ij} q_{ik}(r_j) + \delta_{ij}(t - r_j) \right) + w_j$$

Hence it suffices again to prove (4), which has been proved previously. ◀

The following lemma guarantees that, by the rejection policy, the algorithm rejects at most a small fraction of the total job weight.

▶ **Lemma 4.** *For the set $\mathcal{R}$ of jobs rejected by the algorithm $\mathcal{A}$ it holds that $\sum_{j \in \mathcal{R}} w_j \leq \epsilon_r \sum_{j \in \mathcal{J}} w_j$.*

Proof. Each job $j \in \mathcal{J}$ dispatched to machine $i \in \mathcal{M}$ may increase only the counter v_k of the job $k \in \mathcal{J}$ that was executed on i at r_j. In other words, each job j may be charged to at most one other job. Besides, we reject a job k the first time where $v_k > \frac{w_k}{\epsilon_r}$, meaning that the total weight of jobs charged to k is at least $\frac{w_k}{\epsilon_r}$. Hence, the total weight of rejected jobs is at most an ϵ_r-fraction of the total weight of all jobs in the instance. ◀

▶ **Theorem 5.** *Given any $\epsilon_s > 0$ and $\epsilon_r \in (0,1)$, $\mathcal{A}$ is a $(1+\epsilon_s)$-speed $\frac{2(1+\epsilon_r)(1+\epsilon_s)}{\epsilon_r \epsilon_s}$-competitive algorithm that rejects jobs of total weight at most $\epsilon_r \sum_{j \in \mathcal{J}} w_j$.*

Proof. By Lemma 3, the proposed dual variables constitute a feasible solution for the dual program. By definition, the algorithm $\mathcal{A}$ uses for any machine at any time a factor of $1 + \epsilon_s$ higher speed than that of the adversary. By Lemma 4, $\mathcal{A}$ rejects jobs of total weight at most $\epsilon_r \sum_{j \in \mathcal{J}} w_j$. Hence, it remains to give a lower bound for the dual objective based on the proposed dual variables.

We denote by $F_j^{\mathcal{A}}$ the flow-time of a job $j \in \mathcal{J} \setminus \mathcal{R}$ in the schedule of $\mathcal{A}$. By slightly abusing the notation, for a job $k \in \mathcal{R}$, we will also use $F_k^{\mathcal{A}}$ to denote the total time passed after r_k until deciding to reject a job k, that is, if k is rejected at the release of the job $j \in \mathcal{J}$ then $F_k^{\mathcal{A}} = r_j - r_k$. Denote by j_k the job released at the moment we decided to reject k, i.e., for the counter v_k before the arrival of job j_k we have that $w_k/\epsilon_r - w_{j_k} < v_k < w_k/\epsilon_r$.

Let Δ_j be the total increase in the flow-time caused by the arrival of the job $j \in \mathcal{J}$, i.e., $\Delta_j = \Delta_{ij}$, where $i \in \mathcal{M}$ is the machine to which j is dispatched by $\mathcal{A}$. By the definition of

λ_j's, we have

$$\sum_{j\in\mathcal{J}}\lambda_j \geq \frac{\epsilon_r}{1+\epsilon_r}\left(\sum_{j\in\mathcal{J}}\Delta_j + \sum_{k\in\mathcal{R}}\left(q_{ik}(r_{j_k})\sum_{\ell\in Q_i(r_{j_k})\cup\{k\}:\ell\neq j_k} w_\ell\right)\right)$$
$$= \frac{\epsilon_r}{1+\epsilon_r}\left(\sum_{j\in\mathcal{J}} w_j F_j^{\mathcal{A}} + \sum_{j\in\mathcal{J}}\left(w_j\sum_{k\in D_j} q_{ik}(r_{j_k})\right)\right)$$

where the inequality comes from the fact that if $\delta_{ij} > \delta_{ik}$ then in the prediction part of the running job k at r_j we charge the quantity $w_j p_k$ instead of $w_j q_k(r_j)$ which is the real contribution of k to the weighted flow-time of job j. By the definition of $\gamma_i(t)$'s, we have

$$\sum_{i\in\mathcal{M}}\int_0^\infty \gamma_i(t)dt = \frac{\epsilon_r}{1+\epsilon_r}\left(\sum_{i\in\mathcal{M}}\int_0^\infty\sum_{\ell\in Q_i(t)} w_\ell dt + \sum_{i\in\mathcal{M}}\int_0^\infty\sum_{\ell\in U_i(t)} w_\ell dt\right)$$
$$= \frac{\epsilon_r}{1+\epsilon_r}\left(\sum_{j\in\mathcal{J}} w_j F_j^{\mathcal{A}} + \sum_{j\in\mathcal{J}}\left(w_j\sum_{k\in D_j} q_{ik}(r_{j_k})\right)\right)$$

since the set $Q_i(t)$ contains the pending jobs at time t dispatched on machine i, while each job $j\in\mathcal{J}$ appears by definition in $U_i(t)$ for $\sum_{k\in D_j} q_{ik}(r_{j_k})$ time after its completion or rejection.

Therefore, the proposed assignment for the dual variables leads to the following value of the dual objective

$$\sum_{j\in\mathcal{J}}\lambda_j - \frac{1}{1+\epsilon_s}\sum_{i\in\mathcal{M}}\int_0^\infty \gamma_i(t)dt$$
$$\geq \frac{\epsilon_r\epsilon_s}{(1+\epsilon_r)(1+\epsilon_s)}\left(\sum_{j\in\mathcal{J}} w_j F_j^{\mathcal{A}} + \sum_{j\in\mathcal{J}}\left(w_j\sum_{k\in D_j} q_{ik}(r_{j_k})\right)\right)$$
$$\geq \frac{\epsilon_r\epsilon_s}{(1+\epsilon_r)(1+\epsilon_s)}\sum_{j\in\mathcal{J}} w_j F_j^{\mathcal{A}} \geq \frac{\epsilon_r\epsilon_s}{(1+\epsilon_r)(1+\epsilon_s)}\sum_{j\in\mathcal{J}\setminus\mathcal{R}} w_j F_j^{\mathcal{A}}$$

Since the objective value of our linear program is at most twice the value of an optimal non-preemptive schedule, the theorem follows. ◀

4 ℓ_k-norm on Unrelated Machines

In this section, we study the objective of minimizing the weighted ℓ_k-norm of flow-times. Let $\epsilon_s > 0$ and $0 < \epsilon_r < 1$ be the speed augmentation and the rejection fraction of our algorithm, respectively. For each machine $i\in\mathcal{M}$, job $j\in\mathcal{J}$ and time $t\geq r_j$, we introduce a binary variable $x_{ij}(t)$ which indicates if j is processed on i at time t. We consider the following linear programming formulation. Note that the optimal objective value of this linear program is at most $\frac{4(20k)^{k+3}}{\epsilon_s^{k+1}}$ times the total weighted k-th power of flow-time of jobs in an optimal

preemptive schedule.

$$\min \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\int_{r_j}^{\infty}\frac{2(20k)^{k+3}}{\epsilon_s^{k+1}}\delta_{ij}\left[(t-r_j)^k+p_{ij}^k\right]x_{ij}(t)dt$$

$$\sum_{i\in\mathcal{M}}\int_{r_j}^{\infty}\frac{x_{ij}(t)}{p_{ij}}dt\geq 1\quad\forall j\in\mathcal{J}$$

$$\sum_{j\in\mathcal{J}}x_{ij}(t)\leq 1\quad\forall i\in\mathcal{M},t$$

$$x_{ij}(t)\in\{0,1\}\quad\forall i\in\mathcal{M},j\in\mathcal{J},t\geq r_j$$

After relaxing the integrality constraints, we get the following dual program.

$$\max \sum_{j\in\mathcal{J}}\lambda_j-\sum_{i\in\mathcal{M}}\int_0^{\infty}\gamma_i(t)dt$$

$$\frac{\lambda_j}{p_{ij}}-\gamma_i(t)\leq\frac{2(20k)^{k+3}}{\epsilon_s^{k+1}}\delta_{ij}\left[(t-r_j)^k+p_{ij}^k\right]\quad\forall i\in\mathcal{M},j\in\mathcal{J},t\geq r_j$$

$$\lambda_j,\gamma_i(t)\geq 0\quad\forall i\in\mathcal{M},j\in\mathcal{J},t$$

The algorithm follows the same ideas as the one in the previous section for the objective of minimizing the total weighted flow-time. Each job is immediately dispatched to a machine upon its arrival. Recall that $Q_i(t)$ is the set of pending jobs at time t dispatched to machine $i\in\mathcal{M}$, i.e., the set of jobs dispatched to i that have been released but not yet completed and have not been rejected at t. Our *scheduling policy* for each machine $i\in\mathcal{M}$ is the same as the previous one: at each time t when the machine i becomes idle or has just completed or interrupted some job, we start executing on i the job $j\in Q_i(t)$ of largest density, i.e., $j=\text{argmax}_{j'\in Q_i(t)}\{\delta_{ij'}\}$. In case of ties, we select the job that arrived the earliest.

When a machine $i\in\mathcal{M}$ starts executing a job $u\in\mathcal{J}$, a counter v_u associated to job u is initialized to zero. Each time when a job $j\in\mathcal{J}$ with $\delta_{ij}>\delta_{iu}$ is released during the execution of u and j is dispatched to i, we increase v_u by w_j. Then, the *rejection policy* is the following: we interrupt the execution of the job k and we reject it the first time where $v_u>\frac{w_u}{\epsilon_r}$. As before we define the set of rejected jobs D_j which causes a decrease to the flow time of j and we say that j is *definitively finished* $\sum_{u\in D_j}q_{iu}(r_{j_u})$ time after its completion or rejection. However, j does not appear to the set of pending jobs $Q_i(t)$ for any t after its completion or rejection. Recall that $U_i(t)$ is the set of jobs that have been marked finished at a time before t in machine i but they have not yet been definitively finished at t. For a job $j\in Q_i(t)\cup U_i(t)$, let $F_j(t)$ be the *remaining time* of j from t to to the moment it is definitively finished.

Let Δ_{ij} be the increase in the total weighted k-th power of flow-time occurred in the schedule of our algorithm if we assign a new job $j\in\mathcal{J}$ to machine i, following the above scheduling and rejection policies. Assuming that the job $u\in\mathcal{J}$ is executed on i at time r_j, we have that, if $v_u+w_j\leq\frac{w_u}{\epsilon_r}$ then

$$\Delta_{ij}=w_j\left(q_{iu}(r_j)+\sum_{\substack{a\in Q_i(r_j)\cup\{j\}\setminus\{u\}:\\ \delta_{ia}\geq\delta_{ij}}}p_{ia}\right)^k+\sum_{\substack{a\in Q_i(r_j)\setminus\{u\}:\\ \delta_{ia}<\delta_{ij}}}w_a\left[\left(F_a(r_j)+p_{ij}\right)^k-F_a(r_j)^k\right],$$

otherwise,

$$\Delta_{ij}=w_j\left(\sum_{\substack{a\in Q_i(r_j)\cup\{j\}:\\ \delta_{ia}\geq\delta_{ij}}}p_{ia}\right)^k+\sum_{\substack{a\in Q_i(r_j)\setminus\{u\}:\\ \delta_{ia}<\delta_{ij}}}w_a\left[\left(F_a(r_j)+p_{ij}-q_{iu}(r_j)\right)^k-F_a(r_j)^k\right],$$

where, in both cases, the first term corresponds to the weighted k-th power of the flow-time of job j if it is dispatched to i and the second term corresponds to the change of the weighted k-th power of flow-time for the jobs in $Q_i(r_j)$. Note that, the second case corresponds to the rejection of u and hence we do not have the term $q_{iu}(r_j)$ in the weighted flow-time of j, while the flow-time of each pending job is reduced by $q_{iu}(r_j)$.

Based on the above, we define λ_{ij} as follows. If $\delta_{ij} > \delta_{iu}$ then λ_{ij} equals to

$$\frac{2^k(10k)^k}{\epsilon_s^k}\frac{1+\epsilon_r}{\epsilon_r}w_j p_{ij}^k + \left(1+\frac{\epsilon_s}{5}\right)w_j\left(\sum_{\substack{a\in Q_i(r_j)\cup\{j\}\setminus\{u\}:\\ \delta_{ia}\geq\delta_{ij}}} p_{ia}\right)^k + \sum_{\substack{a\in Q_i(r_j)\setminus\{u\}:\\ \delta_{ia}<\delta_{ij}}} w_a\left[\left(F_a(r_j)+p_{ij}\right)^k - F_a(r_j)^k\right],$$

otherwise, λ_{ij} equals to

$$\frac{2^k(10k)^k}{\epsilon_s^k}\frac{1+\epsilon_r}{\epsilon_r}w_j p_{ij}^k + \left(1+\frac{\epsilon_s}{5}\right)w_j\left(q_{iu}(r_j) + \sum_{\substack{a\in Q_i(r_j)\cup\{j\}\setminus\{u\}:\\ \delta_{ia}\geq\delta_{ij}}} p_{ia}\right)^k + \sum_{\substack{a\in Q_i(r_j)\setminus\{u\}:\\ \delta_{ia}<\delta_{ij}}} w_a\left[\left(F_a(r_j)+p_{ij}\right)^k - F_a(r_j)^k\right].$$

Intuitively, the value of λ_{ij}'s captures the marginal increase of the total weighted k-th power of flow-times due to the arrival of job j and additionally a prediction term. Note that we do not consider the negative quantity $q_{iu(r_j)}$ that appears in the second case of Δ_{ij}.

The *dispatching policy* of the algorithm is the following: dispatch j to the machine $i^* = \text{argmin}_{i\in\mathcal{M}}\{\lambda_{ij}\}$.

It remains to formally define the dual variables. At the arrival of a job $j \in \mathcal{J}$, we set $\lambda_j = \frac{\epsilon_r}{1+\epsilon_r}\min_{i\in\mathcal{M}}\{\lambda_{ij}\}$ and we will never change the value of λ_j again. Define $\gamma_i(t)$ as

$$\gamma_i(t) = \frac{\epsilon_r}{1+\epsilon_r}\left(1+\frac{\epsilon_s}{2}\right)\left(1+\frac{\epsilon_s}{5}\right)\cdot k\sum_{a\in Q_i(t)\cup U_i(t)} w_a F_a(t)^{k-1}$$

Note that $\gamma_i(t)$ is updated during the execution of $\mathcal{A}$. More specifically, given any fixed time t, $\gamma_i(t)$ may increase if a new job j' arrives at any time $r_{j'} \in [r_j, t)$. However, $\gamma_i(t)$ does never decrease in the case of rejection since the jobs remain in $U_i(t)$ for a sufficient time after their completion or rejection.

Using the above definition of the dual variables, the following theorem holds by a quite more technical analysis than that of the previous section for the total weighted flow-time objective.

▶ **Theorem 6.** *Given any $\epsilon_s > 0$ and $\epsilon_r \in (0,1)$, there is a $(1+\epsilon_s)$-speed $O\left(\frac{k^{(k+3)/k}}{\epsilon_r^{1/k}\epsilon_s^{(k+2)/k}}\right)$-competitive algorithm that rejects jobs of total weight at most $\epsilon_r\sum_{j\in\mathcal{J}} w_j$.*

5 Conclusion

In this paper, we presented a generalized model of resource augmentation through the lens of the duality in mathematical programming. The model unifies previous ones and opens

up possibilities for different types of resource augmentation. As shown in the paper, the generalized model can be used to explain the competitiveness of algorithms for certain problems that currently admit no algorithm with performance guarantee even in the resource augmentation context. Besides, an advantage in studying problems in the generalized model is that one can benefit the power of duality-based techniques which have been widely developed for the analysis of approximation and online algorithms. In this context, it would be interesting to consider different problems under the new model. Another interesting question is whether rejection is more powerful than speed. Or, more specifically, can we eliminate the speed augmentation or replace it by a new rejection rule in the presented results? Note that, the power of speed augmentation is that it affects proportionally all jobs, while the difficulty in the rejection case consists in deciding which jobs to reject and how to charge parts of the objective of the non-rejected jobs to the rejected ones.

References

1 Susanne Albers, Lene M. Favrholdt, and Oliver Giel. On paging with locality of reference. *J. Comput. Syst. Sci.*, 70(2):145–175, 2005.

2 S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

3 Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz. On the separation and equivalence of paging strategies. In *Proc. Symposium on Discrete Algorithms*, pages 229–237, 2007.

4 Nikhil Bansal, Ho-Leung Chan, Rohit Khandekar, Kirk Pruhs, B Schicber, and Cliff Stein. Non-preemptive min-sum scheduling with resource augmentation. In *Proc. 48th Symposium on Foundations of Computer Science*, pages 614–624, 2007.

5 Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *J. Comput. Syst. Sci.*, 50(2):244–258, 1995.

6 Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.

7 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.

8 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.

9 Anamitra Roy Choudhury, Syamantak Das, and Amit Kumar. Minimizing weighted ℓ_p-norm of flow-time in the rejection model. In *Proc. 35th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, volume 45, pages 25–37, 2015.

10 Peter J. Denning. The working set model for program behavior. *Commun. ACM*, 11(5):323–333, 1968.

11 Nikhil R. Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms*, 2014.

12 Stefan Dobrev, Rastislav Kralovic, and Dana Pardubská. How much information about the future is needed? In *Proc. 34th Conference on Current Trends in Theory and Practice of Computer Science*, pages 247–258, 2008.

13 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011.

14 Leah Epstein and Rob van Stee. Optimal on-line flow time with resource augmentation. *Discrete Applied Mathematics*, 154(4):611–621, 2006.

15 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Proc. 10th Workshop on Approximation and Online Algorithms*, pages 173–186, 2012.

16 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, 2014.

17 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive flow time algorithms for polyhedral scheduling. In *Proc. 56th Symposium on Foundations of Computer Science*, pages 506–524, 2015.

18 Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *Proc. 55th Symposium on Foundations of Computer Science*, 2014.

19 Sungjin Im, Shi Li, Benjamin Moseley, and Eric Torng. A dynamic programming framework for non-preemptive scheduling problems on multiple machines [extended abstract]. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms*, pages 1070–1086, 2015.

20 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

21 Hans Kellerer, Thomas Tautenhahn, and Gerhard J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. *SIAM J. Comput.*, 28(4):1155–1166, 1999.

22 Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30(1):300–317, 2000.

23 Benjamin Moseley, Kirk Pruhs, and Cliff Stein. The complexity of scheduling for p-norms of flow and stretch - (extended abstract). In *Proc. Integer Programming and Combinatorial Optimization*, pages 278–289, 2013.

24 Cynthia A Phillips, Clifford Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

25 Prabhakar Raghavan. A statistical adversary for on-line algorithms. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:79–83, 1992.

26 Nguyen Kim Thang. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Proc. 21st European Symposium on Algorithms*, pages 755–766, 2013.

27 David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.