Online Non-preemptive Scheduling in a Resource Augmentation Model based on Duality

Abhinav Srivastav[1,2], Giorgio Lucarelli[3], Nguyen Kim Thang[4], Denis Trystram[1]

[1] LIG, UMR 5217, Université Grenoble-Alpes
{abhinav.srivastav,denis.trystram}@imag.fr
[2] Verimag, Université Grenoble-Alpes
[3] INRIA Grenoble Rhône-Alpes
giorgio.lucarelli@inria.fr
[4] IBISC, Université d'Evry
thang@ibisc.fr

Mots-clés : *online scheduling, average weighted flow-time, primal-dual.*

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. In this paper, we are interested in studying the resource augmentation model that compares online algorithms to a weaker adversary. Kalyanasundaram and Pruhs [7] proposed a *speed augmentation* model, where an online algorithm is compared against an adversary with slower processing speed. Phillips et al. [9] proposed the *machine augmentation* model in which the algorithm has more machines than the adversary. Recently, Choudhury et al. [4] introduced the *rejection model* where an online algorithm is allowed to discard a small fraction of jobs.

In this context, we study the problem of online non-preemptive scheduling a set of jobs on unrelated machines in order to minimize the average weighted time a job remains in the system (average weighted flow-time). This is a well representative hard problem since a strong lower bound of $\Omega(\sqrt{n})$ exists even for the offline unweighted version of the problem on a single machine [8], 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 [3]. Moreover, in contrast to the preemptive case, we can show 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. In this paper, we present a competitive algorithm in a model which combines speed augmentation and the rejection model.

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., $\sum_{j\in\mathcal{J}} w_j F_j$.

In what follow, 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 $P = \max_{j,j'\in\mathcal{J}}\{p_j/p_{j'}\}$.

Related Work. 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 [1, 10]. For identical machines, Phillips et al. [9] gave a constant competitive algorithm that uses $m \log P$ machines (recall that the adversary uses m machines). Moreover, an $O(\log n)$-machine $O(1)$-speed algorithm that returns the optimal schedule has been presented in [9] for the unweighted flow-time objective. Epstein and van Stee [5] 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. [2] gave a 12-speed $(2+\epsilon)$-approximation polynomial time algorithm. Recently, Im et al. [6] gave a $(1+\epsilon)$-speed $(1+\epsilon)$-approximation quasi-polynomial time algorithm for the setting of identical machines.

2 Scheduling to Minimize Total Weighted Flow-time

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 based on two lower bounds on the flow-time of each job $j \in \mathcal{J}$, namely its fractional flow-time ($\int_{r_j}^{\infty} \frac{1}{p_{ij}}(t - r_j)x_{ij}(t)dt$) and its processing time ($p_{ij} = \int_{r_j}^{\infty} x_{ij}(t)dt$). After relaxing the integrality constraints, we get also the corresponding dual program.

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

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

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

$$x_{ij}(t) \in \{0, 1\}$$

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

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

$$\lambda_j, \gamma_i(t) \geq 0$$

We will interpret the resource augmentation models in the above primal and dual programs as follows. In speed augmentation, 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. 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

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

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. 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 or 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 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 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. 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

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

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 thus 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 to guide the dispatching policy. The charges have to be distributed in a consistent manner to the assignment decisions of jobs to machines in the past. 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 $\frac{w_j}{\epsilon_r}p_{ij}$. 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

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

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 track 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. Let j_k be the job released at the moment we reject a job $k \in \mathcal{R}$. We say that a job $j \in \mathcal{J}$ 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 $i \in \mathcal{M}$, are already completed or rejected at a time before t, but 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. Based on these definitions, the following theorem holds.

Theorem 1 *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 $\epsilon_r \sum_{j \in \mathcal{J}} w_j$.*

Références

[1] 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.

[2] 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.

[3] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *Proc. of the ACM symposium on Theory of computing*, pages 84–93, 2001.

[4] 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.

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

[6] 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.

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

[8] 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.

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

[10] 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.