Online Min-Sum Flow Scheduling with Rejections

Giorgio Lucarelli, Kim Nguyen, Abhinav Srivastav, Denis Trystram

Online Min-Sum Flow Scheduling with Rejections

Giorgio Lucarelli * Nguyen Kim Thang † Abhinav Srivastav ‡
Denis Trystram *

1 Introduction

A well-identified issue in online computation is the weakness of the worst case paradigm which underestimates the performance of an online algorithm. Many algorithms, which perform well in real-world, are known to admit a mediocre theoretical guarantee. Conversely, many theoretical sound algorithms behave poorly even in simple practical settings. The need of more accurate models is considered as a high importance in algorithmic community. Over the last two decades, several models have been proposed in this direction. We are interested in resource augmentation models in the context of scheduling. Kalyanasundaram and Pruhs [6] proposed the *speed augmentation* model, where an online algorithm is compared against an adversary with slower processing speed, while Phillips et al. [8] proposed the *machine augmentation* model in which the algorithm uses more machines than the adversary. Choudhury et al. [4] introduced the *rejection* model where an online algorithm is allowed to discard a small fraction of jobs.

In this paper, we study the problems of preemptive and non-preemptive online scheduling of jobs on unrelated machines in order to minimize the average time a job remains in the system. Both problems are known to be non-approximable by a constant factor [2, 5]. However, the preemptive variant has been extensively studied under the different resource augmentation models (see for example [1, 3] and the references therein). On the other hand, the non-preemptive variant is much less explored. An $O(\frac{1}{\epsilon})$-competitive algorithm has been presented in [7] for the non-preemptive average flow-time minimization problem on a set of unrelated machines if both an ϵ-speed augmentation is used and an ϵ-fraction of jobs is rejected. We are interested here in exploring the power of the rejection model and, mainly, in eliminating the need for speed augmentation in the latter result. On the road to this, we show how to replace speed augmentation with rejection in the preemptive variant. Our analysis is based on the dual-fitting paradigm.

Problem definition. We are given a set $\mathcal{M}$ of m unrelated machines where 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, while if j is executed on machine $i \in \mathcal{M}$ then it has a *processing time* p_{ij}. Moreover, each job has to be dispatched to one machine at its arrival and migration is not allowed. Given a schedule, let C_j denote the *completion time* of the job j. The *flow-time* of j 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 schedule that minimizes the total flow-times of all jobs, i.e., $\sum_{j \in \mathcal{J}} F_j$.

*{giorgio.lucarelli,denis.trystram}@imag.fr. Univ. Grenoble Alpes, LIG, INRIA, France
†thang@ibisc.fr. IBISC, University of Evry, France
‡abhinav.srivastav@univ-grenoble-alpes.fr. Verimag, Univ. Grenoble Alpes, France

2 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 relaxation and the corresponding dual program.

$$\min \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\int_{r_j}^{\infty}\frac{1}{p_{ij}}(t-r_j+p_{ij})x_{ij}(t)dt$$
$$\sum_{i\in\mathcal{M}}\int_{r_j}^{\infty}\frac{x_{ij}(t)}{p_{ij}}dt \geq 1 \quad \forall j\in\mathcal{J}$$
$$\sum_{j\in\mathcal{J}} x_{ij}(t) \leq 1 \quad \forall i\in\mathcal{M}, t$$
$$x_{ij}(t)\in[0,1]$$

$$\max \sum_{j\in\mathcal{J}}\lambda_j - \sum_{i\in\mathcal{M}}\int_0^{\infty}\gamma_i(t)dt$$
$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq (t-r_j+p_{ij}) \quad \forall i\in\mathcal{M}, j\in\mathcal{J}, t\geq r_j$$
$$\lambda_j, \gamma_i(t) \geq 0$$

We will interpret the rejection model in the above primal and dual programs as follows. We assume that the algorithm is allowed to reject a set $\mathcal{R}$ of jobs. This corresponds to sum up in the primal objective only on the set of the non-rejected jobs. Hence, applying the concept of weak duality, the competitive ratio of an algorithm that rejects the jobs in $\mathcal{R}$ is at most:

$$\frac{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}\setminus\mathcal{R}}\int_{r_j}^{\infty}\frac{1}{p_{ij}}(t-r_j+p_{ij})x_{ij}(t)dt}{\sum_{j\in\mathcal{J}}\lambda_j - \sum_{i\in\mathcal{M}}\int_0^{\infty}\gamma_i(t)dt}$$

3 Preemptive Scheduling

The algorithm. 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, i.e., the set of jobs dispatched to i that have been released but not yet completed or rejected at t. Let $q_{ij}(t)$ be the *remaining processing time* at time t of a job j that is dispatched to machine i. For each machine i, our *scheduling policy* is the following: at each time t we execute on i the job $j \in Q_i(t)$ with the smallest remaining processing time in $Q_i(t)$. In case of ties, we select the job that arrived the earliest. Moreover, we maintain a counter c_i (initialized to 0) for each machine i. Every time a job is dispatched to i, c_i is increased by 1. Then, the *rejection policy* is the following: given an arbitrary small constant $\epsilon \in (0,1)$, whenever c_i reaches $1/\epsilon + 1$, we reject the job with the largest remaining processing time from $Q_i(t)$ and reset c_i to 0. Note that the rejected job does not immediately disappear from the system. We say that a job ℓ is *definitively rejected* at time $t + \sum_{j\in Q_i(t)} q_{ij}(t) + q_{i\ell}(t)$, that is at the time that it supposed to be completed. We denote by $R_i(t)$ the set of jobs dispatched to i that are rejected but not yet definitively at time t. Let Δ_{ij} be the increase in the total flow-time occurred in the schedule of our algorithm following the above scheduling and rejection policies if we decide to dispatch a new job j to machine i. Then, the *dispatching policy* is the following: we dispatch j to the machine where Δ_{ij} is minimum.

Dual variables. Based on the dispatching policy, we set $\lambda_j = \min_i \Delta_{ij}$. Let $W_i(t)$ be the total number of jobs dispatched to machine i that are either pending or not yet definitively rejected until t, i.e., $W_i(t) = |Q_i(t)| + |R_i(t)|$. We set $\gamma_i(t) = W_i(t)/(1+\epsilon)$. Based on this definition, we can guarantee that, given any fixed time t, $\gamma_i(t)$ does not decrease due to rejections since the jobs remain in $R_i(t)$ for sufficient time after their rejection. Then, the following theorem holds.

Theorem 1 *Given any $\epsilon \in (0,1)$, there is an $O(\frac{1}{\epsilon})$-competitive algorithm that rejects at most an ϵ-fraction of jobs.*

4 Non-preemptive Scheduling

The algorithm. We enhance the algorithm presented in the previous section in order to adapt it to a non-preemptive environment where a job is considered to be successfully executed only if its execution is performed without any interruption. The *scheduling policy* for each machine i is the following: at each time t when i is idle, we start executing on i the job that has the smallest processing time in $Q_i(t)$; in case of ties, we select the job that arrived the earliest. We use two rejection rules. The *first rejection policy* is identical with the preemptive case. In order to define the second rejection rule, we maintain another counter v_j for each job j which is initialized to 0 when the execution of j begins. This counter is increased by one each time a new job is dispatched to machine i while j is executing on i. Then, the *second rejection policy* is the following: we reject the job j when $v_j = 1/\epsilon$. The *dispatching policy* is based again on the increase in the total flow-time Δ_{ij} in the same vein as for the preemptive case.

Dual variables. We set $\lambda_j = \epsilon \cdot \min_i \Delta_{ij}$. As before, we define the set $R_i(t)$ of jobs that have been rejected due to the first rejection policy but not yet definitively. For each job j, let D_j denote the set of jobs that are rejected due to the second rejection policy after r_j and before its completion or rejection. Let j_k denote the job released when the job k is rejected due to the second rejection policy. We say that a job j dispatched to machine i 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 and are already completed or rejected due to the second rejection policy but not yet definitively finished at time t. Let $W_i(t)$ be the total number of jobs dispatched to machine i that are pending in $Q_i(t)$, $R_i(t)$ and $U_i(t)$, i.e. $W_i(t) = |Q_i(t)| + |R_i(t)| + |U_i(t)|$. We set $\gamma_i(t) = \frac{\epsilon}{1+\epsilon} W_i(t)$. Based on these definitions, the following theorem holds.

Theorem 2 *Given any $\epsilon \in (0, 1)$, there is an $O(\frac{1}{\epsilon^2})$-competitive algorithm that rejects at most a 2ϵ-fraction of jobs.*

References

[1] S. Anand, N. Garg and A. Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA* , pages 1228–1241, 2012.

[2] C. Chekuri, S. Khanna, and A. Zhu. Algorithms for minimizing weighted flow time. In *STOC*, pages 84–93, 2001.

[3] A. R. Choudhury, S. Das and A. Kumar. Minimizing weighted L_p-norm of flow-time in the rejection model. In *FSTTCS*, vol. 45 of LIPIcs, pages 25–37, 2015.

[4] A. R. Choudhury, S. Das, N. Garg and A. Kumar. Rejecting jobs to minimize load and maximum flow-time. In *SODA*, pages 1114–1133, 2015.

[5] N. Garg and A. Kumar. Minimizing average flow-time : Upper and lower bounds. In *FOCS*, pages 603–613, 2007.

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

[7] G. Lucarelli, N. K. Thang, A. Srivastav and D. Trystram. Online non-preemptive scheduling in a resource augmentation model based on duality. In *ESA*, vol. 57 of LIPIcs 2016.

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