On-Line Bicriteria Interval Scheduling

Fabien Baille, Evripidis Bampis, Christian Laforest, and Nicolas Thibault

LaMI, CNRS UMR 8042, Université d'Evry,
Tour Evry 2, 523, Place des Terrasses 91000 Evry, France
{fbaille,bampis,laforest,nthibaul}@lami.univ-evry.fr

Abstract. We consider the problem of scheduling a sequence of *intervals* revealed *on-line* one by one in the order of their release dates on a set of k identical machines. Each interval i is associated with a processing time p_i and a pair of arbitrary weights (w_i^A, w_i^B) and may be *scheduled* on one of the k identical machines or *rejected*. The objective is to determine a valid schedule maximizing the sum of the weights of the scheduled intervals for each coordinate. We first propose a generic on-line algorithm based on the combination of two monocriteria on-line algorithms and we prove that it gives rise to a pair of competitive ratios that are function of the competitive ratios of the monocriteria algorithms in the input. We apply this technique to the special case where $w_i^A = 1$ and $w_i^B = p_i$ for every interval and as a corollary we obtain a pair of constant competitive ratios.

We consider the problem of scheduling in an on-line context a set of n intervals on k identical machines. An interval i is defined as a tuple of five positive real numbers $(r_i, p_i, d_i, w_i^A, w_i^B)$, where r_i denotes the *release date*, p_i the *processing time*, $d_i = r_i + p_i$ the *deadline* and w_i^A and w_i^B two arbitrary *weights*. We consider the following on-line context: Intervals arrive (are *revealed*) one by one in increasing order of their release dates, i.e. $r_1 \leq r_2 \leq \cdots \leq r_i \leq \cdots$, and they are not known before they are revealed. A revealed interval must either be *served* or *rejected*. An interval i is said to be *served* or *accepted* if it is alloted exclusively and without interruption (preemption is not allowed) to one of the k machines from date r_i to date d_i. Note that the acceptance of an interval may lead to the *interruption* of already scheduled intervals. A schedule O is *valid* if every served interval is scheduled at most once and if at each date every machine schedules at most one interval. There are two objective functions that we call the weight $W_A(O)$, defined as the sum of the first-coordinate-weights w_i^A of the accepted intervals, and the weight $W_B(O)$, corresponding to the sum of the second-coordinate-weights w_i^B of the accepted intervals in O. Note that if an interval is rejected or scheduled and interrupted later before its deadline, it is definitely lost and no gain is obtained from it for none of the metrics. In this model, we search for a solution/schedule that simultaneously maximizes the two objectives W_A and W_B. The particular weight function $w_i^A = 1$ (resp. $w_i^B = p_i$) corresponds to the well known SIZE (resp. PROPORTIONAL WEIGHT) problems.

Competitive ratio. In order to analyze the performance of an on-line algorithm, we use the notion of *competitive ratio* [4, 7]. Let $\sigma_1, \cdots, \sigma_n$ be any on-line sequence. For every i, $1 \leq i \leq n$, let $A(\sigma_1, \cdots, \sigma_i)$ be the schedule returned by the algorithm A at *step* i, i.e. when the first i intervals are revealed, and let O_i^* be an optimal schedule of the set $\{\sigma_1, \cdots, \sigma_i\}$ for some metric C. Then A is said to be ρ-competitive for the metric C if, for all i, $1 \leq i \leq n$, this inequality holds:

$$\rho C(A(\sigma_1, \cdots, \sigma_i)) \geq C(O_i^*)$$

For our bicriteria problem, an algorithm A is said to be (ρ, μ)-competitive if it is *simultaneously* ρ-competitive for W_A and μ-competitive for W_B.

Previous works. To the best of our knowledge, this is the first work considering the simultaneous maximization of two different weight functions in an on-line context. Nevertheless, the off-line version of the bicriteria problem has been treated in [2] where a $(\frac{k}{r}, \frac{k}{k-r})$-approximation algorithm $(1 \leq r < k)$ has been proposed. On the contrary, the monocriteria problems have been extensively studied for both the off-line and the on-line versions. In particular, the off-line versions are polynomial (see Faigle and Nawijn [6] for the SIZE and Carlisle and Lloyd [5] or Arkin and Silverberg [1] for the WEIGHT problems). In the on-line context, the algorithm *GOL* of Faigle and Nawijn [6] is optimal for the SIZE problem. For the WEIGHT problem, there is a series of works going from the paper of Woeginger, in [8], who proposed a 4-competitive algorithm for the PROPORTIONAL WEIGHTS problem in a single machine system, to the paper of Bar-Noy et al. [3] who proposed the *LR* algorithm which is $\frac{2}{1-2\delta}$-competitive for the PROPORTIONAL WEIGHT problem in a different model than ours (instead of k machines, they consider a continuous channel where an interval requires less than a portion δ of the total channel).

Outline of the paper. In Section 1, we describe a generic on-line algorithm for the simultaneous maximization of two weight functions W_A and W_B. We prove that it is a $(\frac{k}{r}\rho, \frac{k}{k-r}\mu)$-competitive algorithm, for $1 \leq r \leq k$, where ρ and μ are the competitive ratios of the corresponding monocriteria algorithms. However, up to our knowledge, no on-line algorithm is available for the general WEIGHT problem. So, we focus, in Section 2, on the special case of the *size* and *proportional weights* metrics. We combine the algorithms *GOL* of [6] for the *size* criterion and of *LR* of [3] for the *proportional weights* criterion in our generic method. We thus propose a bicriteria on-line algorithm and we prove that it induces a pair of constant competitive ratios for this bicriteria case. Finally, we prove in the appendix the competitiveness of *LR*.

1 Our Generic Bicriteria Algorithm

In this section, we describe our generic bicriteria on-line algorithm. It uses as subroutines two on-line monocriteria algorithms having the following structure.

Structure of the monocriteria algorithms. At the release date r_i of a new interval σ_i, any on-line monocriterion algorithm can be split into two main stages. In the first one, called the *interrupting stage*, a set of already scheduled intervals are selected to be interrupted at time r_i. This set can potentially be empty meaning that no interval is interrupted when the algorithm considers σ_i. The second stage is the *scheduling stage*. Here, the algorithm can either reject the interval σ_i or schedule it on one of the available machines.

The rough idea of our generic algorithm is the following: it simulates the execution of two algorithms, say A for the maximization of the weight W_A and B for the maximization of the weight W_B on r and $k-r$ machines, respectively. By doing this, it builds its own interrupting (resp. scheduling) stage from the corresponding interrupting (resp. scheduling) stage of the input algorithms.

1.1 The Algorithm AB^k

We consider the i-th step of an arbitrary algorithm for the WEIGHT problem, i.e. the step at which interval σ_i is released. For any algorithm ALG and for every execution step i of this algorithm, let $\mathcal{O}_{i_1}(ALG)$ (resp. $\mathcal{O}_{i_2}(ALG)$) be the schedule given by ALG after the execution of its *interrupting* (resp. *scheduling*) stage of step i.

Given two algorithms A for the maximization of the weight W_A and B for the weight W_B, our generic algorithm AB^k is constructed as follows: AB^k builds the final schedule by combining the schedules returned by algorithms A and B when applied on r machines and $k-r$ machines, respectively. For the ease of presentation, we denote by A^r (resp. B^{k-r}) the algorithm A (resp. B) when applied on r (resp. $k-r$) machines. We also call *real* (resp. *virtual*) the machines involved in the algorithm AB^k (resp. A^r and B^{k-r}).

For every execution step i of AB^k, let $\mathcal{R}_{i_1}(AB^k)$ (resp. $\mathcal{R}_{i_2}(AB^k)$) be the set of scheduled intervals after the *interrupting* (resp. *scheduling*) stage of step i on the real machines associated to AB^k.

For every step i of the algorithm A^r (resp. B^{k-r}), let $\mathcal{V}_{i_1}(A^r)$ (resp. $\mathcal{V}_{i_1}(B^{k-r})$) be the set of scheduled intervals after the *interrupting stage* of step i on the r (resp. $k-r$) virtual machines associated to A^r (resp. B^{k-r}), and let $\mathcal{V}_{i_2}(A^r)$ (resp. $\mathcal{V}_{i_2}(B^{k-r})$) be the set of scheduled intervals after the *scheduling stage* of step i on the r (resp. the $k-r$) virtual machines associated to A^r (resp. B^{k-r}).

Algorithm AB^k

Input: k identical machines and an on-line sequence of intervals $\sigma_1, \dots, \sigma_n$.
Output: After each step i ($1 \le i \le n$), a valid schedule $\mathcal{O}_{i_2}(AB^k)$ involving a subset of $\sigma_1, \dots, \sigma_i$ on k real machines.
Step 0: $\mathcal{V}_{0_2}(A^r) = \mathcal{V}_{0_2}(B^{k-r}) = \mathcal{R}_{0_2}(AB^k) = \emptyset$.
Step i (date r_i):

1. The **interrupting stage** of AB^k:
   (a) Execute the *interrupting stage* of A^r (resp. B^{k-r}) on the r (resp. $k-r$) virtual machines associated to A^r (resp. B^{k-r}) by submitting the new interval σ_i to A^r (resp. B^{k-r}). Note that the set of intervals scheduled and not interrupted by A^r (resp. B^{k-r}) is now $\mathcal{V}_{i_1}(A^r)$ (resp. $\mathcal{V}_{i_1}(B^{k-r})$).
   (b) On the k real machines associated to AB^k, interrupt the intervals of $\mathcal{R}_{(i-1)_2}(AB^k)$ such that after this interruption we get:
$$\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r}).$$

2. The **scheduling stage** of AB^k:
   (a) Execute the *scheduling stage* of A^r (resp. B^{k-r}) on the r (resp. $k-r$) virtual machines associated to A^r (resp. B^{k-r}) by serving or rejecting the new interval σ_i.
   (b) On the k real machines associated to AB^k, switch to the appropriate case:
      i. If A^r and B^{k-r} reject σ_i, then AB^k does not schedule (rejects) σ_i. Thus, we have:
      $$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k).$$
      ii. If A^r or B^{k-r} serves σ_i (including the case in which both A^r and B^{k-r} serve σ_i), then AB^k schedules σ_i on any free real machine at time r_i. Thus, we have:
      $$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i\}.$$

1.2 Competitiveness of AB^k

Here, we analyze the competitiveness of AB^k. We start with the following lemma which states that AB^k returns a valid schedule and executes the same set of intervals as the union of A^r and B^{k-r}.

Lemma 1 *For every step i of the algorithm AB^k, the schedule $\mathcal{O}_{i_2}(AB^k)$ is valid and we have:*
$$\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$$

Proof. We prove this lemma by induction on the execution steps i of AB^k.
The basic case (step 0): By definition $\mathcal{V}_{0_2}(A^r) = \mathcal{V}_{0_2}(B^{k-r}) = \mathcal{R}_{0_2}(AB^k) = \emptyset$ and thus, $\mathcal{O}_{i_2}(AB^k)$ is valid and of course $\mathcal{R}_{0_2}(AB^k) = \mathcal{V}_{0_2}(A^r) \cup \mathcal{V}_{0_2}(B^{k-r})$.
The main case (step i): Let us assume that $\mathcal{O}_{(i-1)_2}(AB^k)$ is valid and that $\mathcal{R}_{(i-1)_2}(AB^k) = \mathcal{V}_{(i-1)_2}(A^r) \cup \mathcal{V}_{(i-1)_2}(B^{k-r})$ (assumption of induction).

1. The interrupting stage: We first need to prove that:
   $\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r})$ and that $\mathcal{O}_{i_1}(AB^k)$ is valid.
   (a) By definition AB^k interrupts a subset of intervals of $\mathcal{R}_{(i-1)_2}(AB^k)$ in such a way that:
   $$\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r}) \tag{1}$$
   We have to show that there is always a subset of $\mathcal{R}_{(i-1)_2}(AB^k)$ that can be removed such that the above equality is possible.
   Since $\mathcal{V}_{i_1}(A^r) \subseteq \mathcal{V}_{(i-1)_2}(A^r)$, $\mathcal{V}_{i_1}(B^{k-r}) \subseteq \mathcal{V}_{(i-1)_2}(B^{k-r})$ and given that $\mathcal{R}_{(i-1)_2}(AB^k) = \mathcal{V}_{(i-1)_2}(A^r) \cup \mathcal{V}_{(i-1)_2}(B^{k-r})$ (by the assumption of induction), we have $\mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r}) \subseteq \mathcal{R}_{(i-1)_2}(AB^k)$.
   (b) By definition, AB^k interrupts only intervals scheduled in $\mathcal{O}_{(i-1)_2}(AB^k)$, and by the induction hypothesis, $\mathcal{O}_{(i-1)_2}(AB^k)$ is valid. Thus, $\mathcal{O}_{i_1}(AB^k)$ is clearly valid.

2. The scheduling stage: Now, we have to prove that: $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$ and that $\mathcal{O}_{i_2}(AB^k)$ is valid. By the definition of AB^k, several cases may occur:
   (a) If A^r and B^{k-r} reject σ_i, then AB^k does not schedule σ_i and we have:
      i. $\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k)$ (by the definition of AB^k)
         $= \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r})$ (by (1))
         $= \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$
         (since A^r and B^{k-r} reject σ_i, we have: $\mathcal{V}_{i_1}(A^r) = \mathcal{V}_{i_2}(A^r)$ and $\mathcal{V}_{i_1}(B^{k-r}) = \mathcal{V}_{i_2}(B^{k-r})$)
      ii. $\mathcal{O}_{i_2}(AB^k) = \mathcal{O}_{i_1}(AB^k)$. Thus $\mathcal{O}_{i_2}(AB^k)$ is valid (because in item 1b of this proof, we have already seen that $\mathcal{O}_{i_1}(AB^k)$ is valid).
   (b) If A^r (resp. B^{k-r}) serves σ_i and B^{k-r} (resp. A^r) rejects σ_i, then AB^k schedules σ_i on any free real machine at time r_i. We have:
      i. $\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i\}$ (by the definition of AB^k)
         $= \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i\}$ (by (1))
         $= \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$
         (since A^r (resp. B^{k-r}) serves σ_i and B^{k-r} (resp. A^r) rejects σ_i, we have: $\mathcal{V}_{i_2}(A^r) = \mathcal{V}_{i_1}(A^r) \cup \{\sigma_i\}$ (resp. $\mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i\}$) and $\mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_1}(B^{k-r})$ (resp. $\mathcal{V}_{i_2}(A^r) = \mathcal{V}_{i_1}(A^r)$)).
      ii. Since $\mathcal{O}_{i_1}(AB^k)$ is a valid schedule (by the item 1b of this proof) and $\mathcal{O}_{i_2}(AB^k)$ is built by adding σ_i to $\mathcal{O}_{i_1}(AB^k)$ only once, the only reason for which $\mathcal{O}_{i_2}(AB^k)$ could not be valid would be because σ_i is scheduled by AB^k at time r_i whereas there is no free machine at time r_i, i.e. because there is at least $k+1$ intervals of $\mathcal{R}_{i_2}(AB^k)$ scheduled at time r_i by AB^k. Let us prove that this is impossible. Indeed, since A^r and B^{k-r} build at each time valid schedules, there are at most $r+k-r=k$ intervals of $\mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$ scheduled at time r_i by A^r and B^{k-r}, and thus, there are at most k intervals of $\mathcal{R}_{i_2}(AB^k)$ scheduled at time r_i by AB^k (because we have just proved above that $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$). Thus, $\mathcal{O}_{i_2}(AB^k)$ is a valid schedule.
   (c) If A^r and B^{k-r} serve σ_i, then AB^k schedules σ_i on any idle machine at time r_i and we get:
      i. $\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i\}$ (by the definition of AB^k)
         $= \mathcal{V}_{i_1}(A^r) \cup \mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i\}$ (by (1))
         $= \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$
         (since A^r and B^{k-r} serve σ_i, we have $\mathcal{V}_{i_2}(A^r) = \mathcal{V}_{i_1}(A^r) \cup \{\sigma_i\}$ and $\mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i\}$)
      ii. We prove that $\mathcal{O}_{i_2}(AB^k)$ is valid in the same way as before. □

A direct consequence of Lemma 1 is that AB^k is better than A^r (resp. B^{k-r}) for the weight function that A^r (resp. B^{k-r}) maximizes.

Corollary 1 *Let W_A and W_B be two arbitrary weight functions. For every input sequence $\sigma_1, \ldots, \sigma_n$ and for each step i $(1 \le i \le n)$ of AB^k, we have:*
$$W_A(\mathcal{V}_{i_2}(A^r)) \le W_A(\mathcal{R}_{i_2}(AB^k)) \text{ and } W_B(\mathcal{V}_{i_2}(B^{k-r})) \le W_B(\mathcal{R}_{i_2}(AB^k))$$

Proof. By Lemma 1, for every step i of the algorithm AB^k, we have $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \cup \mathcal{V}_{i_2}(B^{k-r})$ and thus Corollary 1 is valid. □

In the following lemma, we analyze, for any type of weight function W, the competitive ratio of the algorithm A applied on r ($r \leq k$) machines when compared to the optimal schedule on a system of k machines.

Lemma 2 *Let $\sigma_1, \cdots, \sigma_n$ be any on-line sequence of intervals. Let A be an on-line algorithm with competitiveness ρ on r machines ($r \leq k$) and O_k^* (resp. O_r^*) be an optimal schedule of $\sigma_1, \cdots, \sigma_n$ for the weight function W on k (resp. r) machines and O_r be the schedule returned by A_r on $\sigma_1, \cdots, \sigma_n$ on r machines. Then,*

$$W(O_k^*) \leq \tfrac{k}{r}\rho W(O_r)$$

Proof. Since A is ρ-competitive, we have by definition $W(O_r^*) \leq \rho W(O_r)$. If we multiply both sides of this inequality by $\frac{k}{r}$, we get $\frac{k}{r}W(O_r^*) \leq \frac{k}{r}\rho W(O_r)$.

Let O_1 be the schedule composed of the first r machines of O_k^* in the decreasing order of their weights. Since O_1 is an r-machine schedule, its weight is at most $W(O_r^*)$. We thus have:

$$\frac{k}{r}W(O_1) \leq \frac{k}{r}W(O_r^*) \leq \frac{k}{r}\rho W(O_r) \tag{2}$$

Since O_1 is an r-machine-schedule executing the intervals scheduled on the r machines generating the maximum weight in O_k^*, the average weight per machine in O_1 is greater than the average weight per machine in O_k^*. So, we have: $\frac{W(O_k^*)}{k} \leq \frac{W(O_1)}{r}$. Combining this result with (2), we get: $W(O_k^*) \leq \frac{k}{r}\rho W(O_r)$. □

Theorem 1 *Let $\sigma_1, \cdots, \sigma_n$ be any on-line sequence of intervals. If A^r is a ρ-competitive algorithm for the weight function W_A on r machines and B^{k-r} is a μ-competitive algorithm for the weight function W_B on $k-r$ machines, then the algorithm AB^k using A^r and B^{k-r} as subroutines is $(\frac{k}{r}\rho, \frac{k}{k-r}\mu)$-competitive.*

Proof. Let $O_k^*(A)$ be an optimal schedule of $\sigma_1, \ldots, \sigma_n$ on k machines for the weight function W_A and $O_k^*(B)$ be an optimal schedule of $\sigma_1, \ldots, \sigma_n$ on k machines for the weight function W_B. By Lemma 2, we have:

$W_A(O_k^*(A)) \leq \frac{k}{r}\rho W_A(\mathcal{V}_{i_2}(A^r))$ and $W_B(O_k^*(B)) \leq \frac{k}{k-r}\mu W_B(\mathcal{V}_{i_2}(B^{k-r}))$

Moreover, using Corollary 1, we have:

$W_A(O_k^*(A)) \leq \frac{k}{r}\rho W_A(\mathcal{R}_{i_2}(AB^k))$ and $W_B(O_k^*(B)) \leq \frac{k}{k-r}\mu W_B(\mathcal{R}_{i_2}(AB^k))$

Thus AB^k is $(\frac{k}{r}\rho, \frac{k}{k-r}\mu)$-competitive. □

2 Application to the SIZE and the PROPORTIONAL WEIGHT

Given that, to the best of our knowledge, we do not know on-line algorithm with constant competitive ratio for general weight functions, we focus in this section on the particular case where $w_i^A = 1$ and $w_i^B = p_i$ for every $i = 1, \ldots, n$, i.e. for the *size* and *proportional weights* metrics. We first show that the optimal on-line algorithm *GOL* of Faigle and Nawijn [6] can be described following the two-stages structure presented in the previous section. We also present in this form the on-line algorithm LR^k of Bar-Noy et al. [3]. Recall that GOL^k is optimal for the SIZE problem while *LR* deals with *proportional weights* (but for a different model than the one adopted here). Then, we use these algorithms as input of our generic method.

Here is a description of the algorithm GOL^k. It is the original algorithm *GOL* of [6] (using k machines) except that it is split into an interrupting stage and a scheduling stage.

Algorithm GOL^k[6]

At the arrival of interval σ_i do:
Interrupting stage: If there are k served intervals intersecting the date r_i, let σ_{max} be the one with the maximum deadline.
If σ_{max} does not exist (there is a free machine), do not interrupt any interval.
If $d_{max} \geq d_i$ then interrupt σ_{max}.
If $d_{max} < d_i$ then do not interrupt any interval.
Scheduling stage: If an interval has been interrupted (a machine became idle) or if there is a free machine, then schedule σ_i on any free machine. Else, reject σ_i.

We now adapt the algorithm *LR*. In [3], *LR* is described as an algorithm running on a *continuous* channel, where each interval requires a portion (not necessarily contiguous) of this channel. In our model we consider k machines (instead of a continuous channel), and each interval requires exactly one (discrete) machine. That is why we give the description of LR^k (the adaptation of *LR* on a *discrete* model of $k \geq 3$ machines). The proof of its $\frac{2}{1-\frac{2}{k}}$-competitiveness is given in the appendix because Lemma 3 and Theorem 2 are adaptations of the proof of competitiveness of *LR* coming from [3] to our model.

Algorithm LR^k(adaptation of [3])

We define F_t as the set of scheduled intervals containing date t.
When σ_i is revealed do:
Interrupting stage:

- If $|F_{r_i}| < k$, then do not interrupt any interval
- If $|F_{r_i}| = k$, then:
  1. Sort the $k+1$ intervals of $F_{r_i} \cup \{\sigma_i\}$ by increasing order of release dates, if several intervals have the same release date, order them in the decreasing order of their deadlines and let L be the set of the $\lceil \frac{k}{2} \rceil$ first intervals.
  2. Sort the $k+1$ intervals of $F_{r_i} \cup \{\sigma_i\}$ by decreasing order of deadlines (ties are broken arbitrarily) and let R be the set of the $\lfloor \frac{k}{2} \rfloor$ first intervals.

If $\sigma_i \in L \cup R$ then interrupt any interval σ_j of $F_{r_i} - L \cup R$.
Else do not interrupt any interval.

Scheduling stage:

- If $|F_{r_i}| < k$ then schedule σ_i on any free machine.
- If $|F_{r_i}| = k$, then:
    - $*$ If $\sigma_i \in L \cup R$ then schedule σ_i on the machine where σ_j was interrupted.
    - $*$ If $\sigma_i \notin L \cup R$ then reject σ_i.

Theorem 2 *For proportional weights ($w_i = p_i$) and for $k \geq 3$, LR^k is $\frac{2}{1-\frac{2}{k}}$-competitive.*

Recall that GOL^r is an optimal on-line algorithm (i.e. 1-competitive) for the SIZE and LR^{k-r} is an on-line $\frac{2}{1-\frac{2}{k-r}}$-competitive algorithm for the PROPORTIONAL WEIGHTS problem. So, applying Theorem 1, we have:

Corollary 2 *For $k \geq 4$ and for all $1 \leq r \leq k-3$, AB^k applied with $A^r = GOL^r$ and $B^{k-r} = LR^{k-r}$ is $(\frac{k}{r}, \frac{2k}{k-r-2})$-competitive for the* size *and* proportional weights *criteria.*

Note that the parameter r can be tuned in order to make AB^k more precise for one of the objectives. For example, if we set $r = \frac{k-2}{3}$, we obtain a pair of competitive ratios of $(\frac{3}{1-\frac{2}{k}}, \frac{3}{1-\frac{2}{k}}) \leq (6, 6)$ and which tends to $(3, 3)$ for large k. In figure 1, we show all the couples of approximation ratios that our algorithm applied with GOL and LR with $k = 20$ can reach by variations of r.

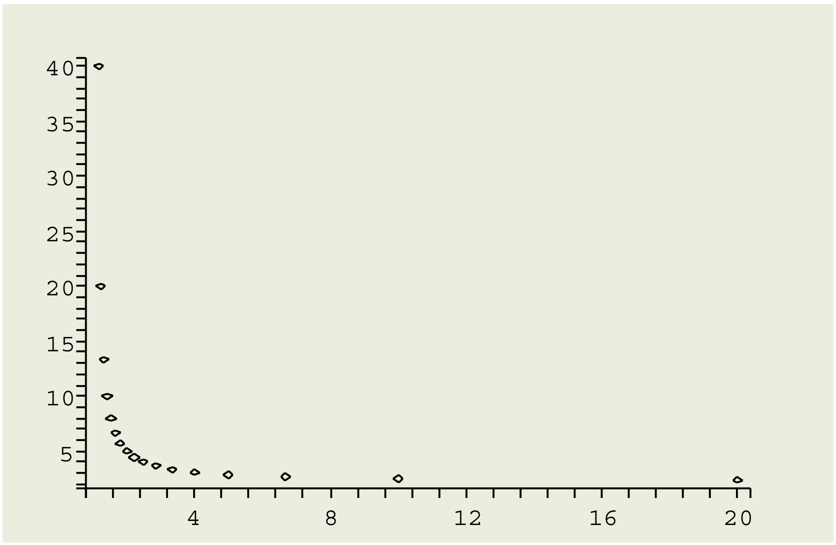

Fig. 1. Competitive ratios for the Weight (Y-axis) and the Size (X-axis) when $k = 20$.

References

1. E. Arkin and B. Silverberg, *Scheduling jobs with fixed start and end times*, Discrete Applied Mathematics, 18 (1987), pp. 1–8.
2. F. Baille, E. Bampis, and C. Laforest, *A note on bicriteria schedules with optimal approximation ratios*, Parallel Processing Letters, 14 (2004), pp. 315–323.
3. A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber, *Bandwidth allocation with preemption*, SIAM J. Comput., 28 (1999), pp. 1806–1828.
4. A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University press, 1998.
5. M. C. Carlisle and E. L. Lloyd, *On the k-coloring of intervals*, Discrete Applied Mathemetics, 59 (1995), pp. 225–235.
6. U. Faigle and M. Nawijn, *Note on scheduling intervals on-line*, Discrete Applied Mathematics, 58 (1995), pp. 13–17.
7. A. Fiat and G. J. Woeginger, *Online algorithms: The state of the art*, LNCS no. 1442, Springer, 1998.
8. G. J. Woeginger, *On-line scheduling of jobs with fixed start and end times*, Theor. Comput. Sci., 130 (1994), pp. 5–16.

Appendix: Proof of Theorem 2

Let $O = LR^k(\sigma_1, \cdots, \sigma_i)$ be the schedule on k machines returned by LR^k on $\sigma_1, \cdots, \sigma_i$. Let T_i^t be the number of intervals of O containing the date t. Let F_i^t be the number of intervals of $\{\sigma_1, \cdots, \sigma_i\}$ containing the date t. For the proof of the Theorem, we need the following result:

Lemma 3 *Using the above notations, the schedule returned by LR^k satisfies:*

$$\forall i,\ \forall t,\ T_i^t \geq \min\{F_i^t,\ \tfrac{k}{2} - 1\}$$

Proof. We proceed by induction on i. For $i = 1$, $\forall t \in [r_1, d_1)$, we have: $T_1^t = F_1^t = 1$ and $\forall t \notin [r_1, d_1)$, $T_1^t = F_1^t = 0$.

Suppose $i > 1$. According to the algorithm, two cases may occur:

First case: $|F_{r_i}| < k$. In this case, σ_i is scheduled by LR^k and no interval is interrupted. If $t \notin [r_i, d_i)$, then the number of scheduled intervals which contain the date t at step i is the same as at step $i - 1$. Thus, we have $T_i^t = T_{i-1}^t$. Moreover, since $t \notin [r_i, d_i)$, we have also $F_i^t = F_{i-1}^t$. So, by replacing T_{i-1}^t by T_i^t and F_{i-1}^t by F_i^t in the induction hypothesis, this particular case is checked. If $t \in [r_i, d_i)$, then since σ_i has been scheduled, we have: $T_i^t = T_{i-1}^t + 1$. By the induction hypothesis, we can rewrite this equation:

$$T_i^t \geq 1 + \min\{F_{i-1}^t,\ \frac{k}{2} - 1\} \tag{3}$$

If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = \frac{k}{2} - 1$, then (3) becomes: $T_i^t \geq 1 + \frac{k}{2} - 1 = \frac{k}{2} > \frac{k}{2} - 1 \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = F_{i-1}^t$, then (3) becomes: $T_i^t \geq 1 + F_{i-1}^t$. But since $t \in [r_i, d_i)$, we have $F_i^t = F_{i-1}^t + 1$. Thus, we have: $T_i^t \geq F_i^t - 1 + 1 = F_i^t \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$.

Second case: $|F_{r_i}| = k$. In this case, three sub-cases may occur: If $\sigma_i \notin L$ and $\sigma_i \notin R$. This means that σ_i is rejected by LR^k. If $t \notin [r_i, d_i)$ then $T_i^t = T_{i-1}^t$ and $F_i^t = F_{i-1}^t$. By replacing T_{i-1}^t by T_i^t and F_{i-1}^t by F_i^t in the induction hypothesis, this particular case is checked. If $t \in [r_i, d_i)$, since $\sigma_i \notin L \cup R$, there are always at least $\lfloor \frac{k}{2} \rfloor$ intervals containing t in O. Thus, $T_i^t \geq \lfloor \frac{k}{2} \rfloor \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$.

If $\sigma_i \in R$ (including the case where σ_i is also in L). This means that σ_i is accepted by LR^k and σ_j is rejected. Then, since σ_j is revealed before σ_i, we have $r_j \leq r_i$. Furthermore, we have $d_j \leq d_i$ otherwise, we would have $\sigma_j \in R$, contradicting the fact that σ_j is interrupted. We have then these cases: For all $t \notin [r_j, d_i)$, we have $F_i^t = F_{i-1}^t$ and $T_i^t = T_{i-1}^t$. Thus, by replacing T_{i-1}^t by T_i^t and F_{i-1}^t by F_i^t in the induction hypothesis, this particular case is checked. For all $t \in [r_j, r_i)$, since $\sigma_j \notin L$, there are at least $\lceil \frac{k}{2} \rceil$ intervals containing the date t. Thus, we have: $T_i^t \geq \lceil \frac{k}{2} \rceil > \min\{F_i^t, \frac{k}{2} - 1\}$. For all $t \in [r_i, d_j)$, we have $T_i^t = T_{i-1}^t$ because σ_j is deleted but σ_i is added. Since $\sigma_j \notin R$, there are at least $\lfloor \frac{k}{2} \rfloor$ intervals containing date t. Thus, we have $T_i^t \geq \lfloor \frac{k}{2} \rfloor \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. For all $t \in [d_j, d_i)$, since σ_i occupies a machine that was free at step $i - 1$ of the algorithm, we have: $T_i^t = T_{i-1}^t + 1$. By the induction hypothesis, we can rewrite this equation:

$$T_i^t \geq 1 + \min\{F_{i-1}^t,\ \frac{k}{2} - 1\} \tag{4}$$

If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = \frac{k}{2} - 1$, then (4) becomes: $T_i^t \geq 1 + \frac{k}{2} - 1 = \frac{k}{2} > \frac{k}{2} - 1 \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = F_{i-1}^t$, then (4) becomes: $T_i^t \geq 1 + F_{i-1}^t$. But since $t \in [r_i, d_i)$, we have $F_i^t = F_{i-1}^t + 1$. Thus, we have: $T_i^t \geq F_i^t - 1 + 1 = F_i^t \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$.

If $\sigma_i \in L$ and $\sigma_i \notin R$. This means that σ_i is accepted by LR^k and σ_j is rejected. By the on-line context, since the last revealed interval is σ_i, all the intervals which do not belong to L have a release date equal to r_i (otherwise they would belong to L). In particular, $\sigma_j \notin L$ because it is interrupted and thus it satisfies $r_j = r_i$. Moreover, by the manner the algorithm builds L, σ_i has also a greater deadline than σ_j (otherwise, $\sigma_j \in L$ and thus it would not be interrupted): $d_j \leq d_i$. We have 3 cases to consider: For all $t \notin [r_i, d_i)$, we have $F_i^t = F_{i-1}^t$ and $T_i^t = T_{i-1}^t$. Thus, by replacing T_{i-1}^t by T_i^t and F_{i-1}^t by F_i^t in the induction hypothesis, this particular case is checked. For all $t \in [r_i, d_j)$, we have $T_i^t = T_{i-1}^t$ because σ_j is deleted but σ_i is added. Since $\sigma_i \notin R$, there are at least $\lfloor \frac{k}{2} \rfloor$ intervals containing date t having a deadline at least d_i. Thus, we have: $T_i^t \geq \lfloor \frac{k}{2} \rfloor \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. For all $t \in [d_j, d_i)$, since σ_i occupies a machine that was free at step $i - 1$ of the algorithm, we have: $T_i^t = T_{i-1}^t + 1$. By the induction hypothesis, we can rewrite this equation:

$$T_i^t \geq 1 + \min\{F_{i-1}^t,\ \frac{k}{2} - 1\} \tag{5}$$

If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = \frac{k}{2} - 1$, then (5) becomes: $T_i^t \geq 1 + \frac{k}{2} - 1 = \frac{k}{2} > \frac{k}{2} - 1 \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. If $\min\{F_{i-1}^t,\ \frac{k}{2} - 1\} = F_{i-1}^t$, then (5) becomes: $T_i^t \geq 1 + F_{i-1}^t$. But since $t \in [r_i, d_i)$, we have $F_i^t = F_{i-1}^t + 1$. Thus, we have: $T_i^t \geq F_i^t - 1 + 1 = F_i^t \geq \min\{F_i^t,\ \frac{k}{2} - 1\}$. We have checked the induction step and thus the lemma. □

proof of Theorem 2: Let O_i^* be the optimal (off-line) weight schedule of $\sigma_1, \ldots, \sigma_i$. Let T_i^{*t} be the number of intervals of the schedule O_i^* containing date t. Let t be a date of the schedule O returned by LR^k on the input sequence $\sigma_1, \cdots, \sigma_i$ and i be a step of the algorithm. If $\min\{F_i^t, \frac{k}{2} - 1\} = F_i^t$ then by Lemma 3, we have $T_i^t \geq F_i^t \geq T_i^{*t}$. Now, let us consider the case in which $\min\{F_i^t, \frac{k}{2} - 1\} = \frac{k}{2} - 1$. Since O_i^* is valid, we have $T_i^{*t} \leq k$. Multiplying both sides by $\frac{1-\frac{2}{k}}{2}$, by remarking that $\frac{k}{2}\left(1 - \frac{2}{k}\right) = \frac{k}{2} - 1$ and by Lemma 3, we obtain:
$\frac{T_i^{*t}}{2}\left(1 - \frac{2}{k}\right) \leq \frac{k}{2}\left(1 - \frac{2}{k}\right) = \frac{k}{2} - 1 \leq T_i^t$.
Thus, we have for all dates t and for all steps i: $\frac{2}{1-\frac{2}{k}} T_i^t \geq T_i^{*t}$. If we sum this inequality for all dates t, we obtain that LR^k is $\frac{2}{1-\frac{2}{k}}$-competitive. □