On-line simultaneous maximization of the size and the weight for degradable intervals schedules

Fabien Baille, Evripidis Bampis, Christian Laforest, Nicolas Thibault

On-line simultaneous maximization of the size and the weight for degradable intervals schedules

Fabien Baille, Evripidis Bampis, Christian Laforest, Nicolas Thibault

Tour Evry 2, LaMI, Université d'Evry, 523 place des terrasses, 91000 EVRY France
{fbaille,bampis,laforest,nthibaul}@lami.univ-evry.fr

Abstract. We consider the problem of scheduling *on-line* a sequence of *degradable* intervals in a set of k identical machines. Our objective is to find a schedule that maximizes *simultaneously* the *Weight* (equal to the sum of processing times) and the *Size* (equal to the number) of the scheduled intervals. We propose a *bicriteria* algorithm that uses the strategies of two monocriteria algorithms (GOL [7], maximizing the *Size* and LR [4], maximizing the *Weight*) and yields two simultaneous constant competitive ratios. This work is an extension of [2] (COCOON'04), where the same model of degradable intervals was investigated in an *off-line* context and the two objectives were considered separately.

In this paper, we consider the problem of scheduling on-line degradable intervals on k identical machines. We define a *degradable* interval σ by a triplet (r, q, d) where r denotes the *release date*, q the *minimal deadline* and d the *deadline* $(r < q \leq d)$. This means that σ is *scheduled* if and only if it is executed from date r to date t $(q \leq t \leq d)$ on one machine. Intuitively, in this model, each interval can be shortened (with respect to the required total execution $[r, d)$). We denote by $[r, t)$ the numerical interval corresponding to the effective execution of a degradable interval σ and by $p(\sigma) = t - r$ its processing time. We define the *weight* $w(\sigma)$ of the effective execution of any interval σ by $w(\sigma) = t_\sigma - r_\sigma$. This means that the weight of an interval σ is equal to its processing time (it is known in the literature as the *proportional weight* model [8]). In our model, we consider on-line sequences of degradable intervals $\sigma_1, ..., \sigma_n$ where the σ_i's are *revealed* one by one in the increasing order of their release dates $(r_1 \leq r_2 \leq \cdots \leq r_n)$, and future intervals are not known in advance.

For any algorithm A, we denote by A^k the version of A running on k identical machines. In our model, an on-line algorithm A^k has to build at each step a *valid* schedule. A schedule is *valid* if and only if for every date t, there is at most one interval on each machine and each interval is scheduled at most once. When a new interval σ_i is revealed (at step i), the algorithm A^k can *reject* it (in this case, it is definitively lost) or *serve* it. In this second case, if the algorithm schedules σ_i on machine j, it interrupts at least the already scheduled interval intersecting σ_i on machine j. The interrupted intervals are definitively lost and no gain is obtained from them for both metrics. Thus, each step i of any on-line algorithm A^k can be decomposed into two stages: First, there is the *interrupting stage* of step i. During this stage, the algorithm interrupts a subset of the already scheduled intervals (note that this subset can be empty). Secondly, there is the *scheduling stage* of step i. During this stage, the algorithm decides if the new interval σ_i is served or rejected, and if it is served, on which machine σ_i is served.

Notation 1 (Schedule $A^k(\sigma_1, \ldots, \sigma_i)$) *Let $\sigma_1, \ldots, \sigma_n$ be any on-line sequence of degradable intervals and let A^k be any algorithm running on k identical machines. For every step i ($1 \leq i \leq n$), we denote by $A^k(\sigma_1, \ldots, \sigma_i)$ the schedule returned by A^k at the end of step i.*

We define the *size* $N(O) = |\{\sigma \in O\}|$ (i.e. the number of scheduled intervals) and the *weight* $W(O) = \sum_{\sigma \in O} w(\sigma)$ (i.e. the weight of scheduled intervals) of any schedule O. Our problem is then to find a schedule which has *size* and *weight* the largest possible. In order to evaluate the quality of a schedule for our measures (the Size *and* the Weight), we use the competitive ratio [5].

Definition 1 (Competitive ratio). *Let $\sigma_1, ..., \sigma_n$ be any on-line sequence of intervals. Let $A^k(\sigma_1, ..., \sigma_i)$ be the schedule on k machines given by an algorithm A^k at step i ($1 \leq i \leq n$) and let O_i^* be the optimal (off-line) schedule on k machines of $\{\sigma_1, \ldots, \sigma_i\}$ for the criterion C (here, $C = W$ or $C = N$). A^k has a competitive ratio of ρ (it is ρ-competitive) for the criterion C if and only if we have:*

$$\forall i,\ 1 \leq i \leq n,\ \rho \cdots C(A^k(\sigma_1, ..., \sigma_i)) \geq C(O_i^*)$$

An algorithm A^k is (ρ, μ)-competitive if it is *simultaneously* ρ-competitive for the Size and μ-competitive for the Weight. In this paper, we propose a $\left(\frac{k}{r}, \frac{4k}{k-r-2}\right)$-competitive algorithm, called AB^k (with $1 \leq r < k$). For example, if we set $r = \frac{k}{2}$ (if $\frac{k}{2} \geq 3$ and k even), AB^k is $\left(2, \frac{8}{1-\frac{4}{k}}\right)$-competitive.

Previous works. The off-line version of the bicriteria non-degradable problem has been treated in [3] where a $(\frac{k}{r}, \frac{k}{k-r})$-approximation algorithm ($1 \leq r < k$) has been proposed. Concerning the monocriteria non-degradable problems, they 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 [7] for the Size and Carlisle and Lloyd [6] or Arkin and Silverberg [1] for the Weight problems). In the on-line context, the algorithm GOL of Faigle and Nawijn [7] is optimal for the Size problem. For the Weight problem, there is a series of works going from the paper of Woeginger [8] to the paper of Bar-Noy et al. [4], who proposed on-line algorithms with constant competitive ratios. Note that the two degradable monocriterion intervals problem has been investigated in [2].

Outline of the paper. In Section 1, we present two monocriterion algorithms (GOL^k [7] and LR^k [4]) in the degradable interval model. Section 2 is devoted to the description and the analysis of our on-line bicriteria algorithm AB^k using GOL and LR as subroutines.

1 Two on-line monocriteria algorithms for the *size* and the *proportional weight* metrics

In this section, we describe and analyze the competitiveness of the algorithm of Faigle and Nawijn [7] and the algorithm of Bar-Noy et al. [4]. We use them as subroutines of our algorithm AB^k (see Section 2) in order to obtain a pair of constant competitive ratios.

The algorithm GOL^k. We describe the algorithm GOL^k of [7] in the degradable interval model by decomposing it into an interrupting stage and a scheduling stage.

Algorithm GOL^k (adaptation of [7])

When a new interval σ_i, defined by (r_i, q_i, d_i), is revealed, choose its effective execution $\sigma_i^q = [r_i, q_i)$ (i.e. each new interval is totally degraded) and 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.
Else, If $d_{max} \geq q_i$ then interrupt σ_{max}.
If $d_{max} < q_i$ then do not interrupt any interval.

Scheduling stage:
If there is a free machine, then schedule σ_i^q on it.
Else, reject σ_i.

Note that the original algorithm of [7] is described for the classical non-degradable interval model. In the following, we denote by GOL_N^k this original version of the algorithm (notice that it is the same algorithm as GOL^k, except that GOL_N^k does not degrade any interval).

Lemma 1 *GOL^k is optimal for the Size in the degradable interval model.*

Proof. Let $\sigma_1, \ldots, \sigma_n$ be an on-line sequence of intervals such that for all i, $1 \leq i \leq n$, σ_i is defined by (r_i, q_i, d_i). Let $\sigma_1^D, \ldots, \sigma_n^D$ be the sequence such that for all i, $1 \leq i \leq n$, $\sigma_i^D = [r_i, q_i)$ (i.e. the sequence version with the intervals totally degraded). Let O_D^* be an optimal schedule of $\sigma_1, \ldots, \sigma_n$ in the degradable interval model and let O_N^* be an optimal schedule of $\sigma_1^D, \ldots, \sigma_n^D$ in the non-degradable interval model. By [2], we know that $N(O_D^*) = N(O_N^*)$. Furthermore, since GOL_N^k is optimal for the *Size* in the non-degradable interval model (See [7]), we have $N(O_N^*) = N(GOL_N^k(\sigma_1^D, \ldots, \sigma_n^D))$. By definition of GOL^k we have $GOL_N^k(\sigma_1^D, \ldots, \sigma_n^D) = GOL^k(\sigma_1, \ldots, \sigma_n)$. If we combine all these equalities, we obtain $GOL^k(\sigma_1, \ldots, \sigma_n) = GOL_N^k(\sigma_1^D, \ldots, \sigma_n^D) = N(O_N^*) = N(O_D^*)$. Thus, GOL^k is optimal for the *Size* in the degradable interval model. □

The algorithm LR^k. We now describe the algorithm LR^k of [4] adapted to our degradable interval model and decomposed into an interrupting stage and a scheduling stage (for all $k \geq 3$).

Algorithm LR^k (adaptation of [4])

We denote by F_t the set of scheduled intervals containing date t. When a new interval σ_i defined by (r_i, q_i, d_i) is revealed, choose the effective execution $\sigma_i^d = [r_i, d_i)$ (i.e. do not degrade any interval) and 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 $F_{r_i} \cup \{\sigma_i^d\}$ 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 $F_{r_i} \cup \{\sigma_i^d\}$ 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 $\sigma_i^d \in L \cup R$, then interrupt any interval σ_j in $F_{r_i} - L \cup R$,
Else do not interrupt any interval.

Scheduling stage:
If $|F_{r_i}| < k$, then schedule σ_i^d on any free machine.
If $|F_{r_i}| = k$ and $\sigma_i^d \in L \cup R$, then schedule σ_i^d on the machine where σ_j has been interrupted.
If $|F_{r_i}| = k$ and $\sigma_i^d \notin L \cup R$, then reject σ_i.

In the following, we show that LR^k is $\left(\frac{4}{1-\frac{2}{k}}\right)$-competitive in the degradable interval model for the *Weight* metric (note that this is very close to 4 when k is large). We first show that the weight of an optimal degradable schedule is no more than twice the weight of an optimal non-degradable schedule.

Lemma 2 *For every set of intervals $\{\sigma_1, \ldots, \sigma_n\}$, let O^{*nd} be an optimal schedule of $\{\sigma_1, \ldots, \sigma_n\}$ for the proportional weight metric in the non-degradable interval model (i.e. $q_i = d_i$), and let O^{*d} be an optimal schedule of $\{\sigma_1, \ldots, \sigma_n\}$ for the same metric in the degradable interval model. We have:*

$$W(O^{*d}) \leq 2W(O^{*nd})$$

Proof. Let $O_1^{*d}, \cdots, O_k^{*d}$ be the k sub-schedules of O^{*d} (O_i^{*d} executes the same intervals at the same dates as machine i of O^{*d}). Thus, we have:

$$W(O^{*d}) = \sum_{i=0}^{k} W(O_i^*)$$

Let $\Gamma_i = \{\sigma_j \in \{\sigma_1, \ldots, \sigma_n\} : \sigma_j \in O_i^{*d}\}$. O_i^{*d} is an optimal schedule of Γ_i in the degradable interval model. Indeed, suppose, by contradiction, that there exists a valid schedule O of Γ_i such that $W(O_i^{*d}) < W(O)$. This means that the valid schedule consisting in the union of the O_j^{*d}'s, except for $j = i$, which is replaced by O, generates a weight greater than O^{*d} and is valid. This contradicts the optimality of O^{*d}.

Let us apply the 2-approximation algorithm for one machine schedules described in [2] separately on each Γ_i ($1 \leq i \leq k$). Let $O_1, \cdots, O_k$ be the obtained schedules. Thus, by Theorem 5 of [2], for each i, we have $W(O_i^{*d}) \leq 2W(O_i)$. We sum the k inequalities and we obtain $W(O^{*d}) \leq 2\sum_{i=1}^{k} W(O_i)$. Moreover, since the 2-approximation algorithm of [2] does not degrade the intervals, the k machine schedule consisting in the union of the O_i's is valid for the non-degradable interval model. This means that $\sum_{i=1}^{k} W(O_i) \leq W(O^{*nd})$. Combining this last inequality with $W(O^{*d}) \leq 2\sum_{i=1}^{k} W(O_i)$ leads to:

$$W(O^{*d}) \leq 2W(O^{*nd})$$

□

Corollary 1 *LR^k is $\left(\frac{4}{1-\frac{2}{k}}\right)$-competitive for the degradable interval model on $k \geq 3$ machines.*

Proof. It is known that LR^k is $(\frac{2}{1-\frac{2}{k}})$-competitive for the non-degradable interval model (from an adaptation of the proof of [4]). Thus, by definition, we have $\frac{2}{1-\frac{2}{k}} W(LR^k(\sigma)) \geq W(O^{*nd})$. By Lemma 2, we have $2W(O^{*nd}) \geq W(O^{*d})$. Combining these two inequalities leads to $\frac{4}{1-\frac{2}{k}} W(LR^k(\sigma)) \geq W(O^{*d})$. This means that LR^k is $\left(\frac{4}{1-\frac{2}{k}}\right)$-competitive for the degradable model. □

2 Our algorithm AB^k

Definition 2 (Cover relation). *Let σ be an interval defined by the triplet (r, q, d). Let $\sigma^1 = [r, t_1)$ and $\sigma^2 = [r, t_2)$ be two valid effective executions of σ (i.e. $q \leq t_1 \leq d$ and $q \leq t_2 \leq d$). We say that σ^1* covers *σ^2 if and only if $\sigma^2 \subseteq \sigma^1$ (i.e. if and only if $t_2 \leq t_1$).*

Definition 3 (Union$^+$ of sets of degraded intervals). *Let $\{\sigma_1, \cdots, \sigma_n\}$ be a set of degradable intervals. For all i, $1 \leq i \leq n$, σ_i is defined by (r_i, q_i, d_i). For all $\sigma_i \in \{\sigma_1, \cdots, \sigma_n\}$, let $\sigma_i^1 = [r_i, t_i^1)$ and $\sigma_i^2 = [r_i, t_i^2)$ be two valid effective executions of σ_i (i.e. $q_i \leq t_i^1 \leq d_i$ and $q_i \leq t_i^2 \leq d_i$). Let $E_1 \subseteq \{\sigma_i^1 : \sigma_i \in \{\sigma_1, \ldots, \sigma_n\}\}$ and $E_2 \subseteq \{\sigma_i^2 : \sigma_i \in \{\sigma_1, \ldots, \sigma_n\}\}$. We define $E_1 \uplus E_2$ the* union$^+$ *of E_1 and E_2 as follows:*

- *$\sigma_i^1 \in E_1 \uplus E_2$ if and only if ($\sigma_i^1 \in E_1$ and $\sigma_i^2 \notin E_2$) or ($\sigma_i^1 \in E_1$ and $\sigma_i^2 \in E_2$ and σ_i^1 covers σ_i^2).*
- *$\sigma_i^2 \in E_1 \uplus E_2$ if and only if ($\sigma_i^2 \in E_2$ and $\sigma_i^1 \notin E_1$) or ($\sigma_i^2 \in E_2$ and $\sigma_i^1 \in E_1$ and σ_i^2 covers σ_i^1).*

Note that the union$^+$ is commutative and it generalizes the usual definition of the union of two non-degradable intervals sets since in that case, $\sigma_i^1 = \sigma_i^2$. Thus, for all σ, E_1 and E_2, if $\sigma \in E_1 \uplus E_2$, then $\sigma \in E_1 \cup E_2$.

As, by definition, the two effective executions of a same interval σ defined by (r, q, d) must start at the same release date r, the one with the smallest execution time is covered by the other.

Note that, to be completely rigorous, we should not define an interval σ_i by (r_i, q_i, d_i), but by (r_i, q_i, d_i, i). Indeed, let us consider the following problematic example. Let $\sigma_i^1 = [r_i, t_i^1)$ be an effective execution of $\sigma_i = (r_i, q_i, d_i)$ and $\sigma_j^1 = [r_j, t_j^1)$ be an effective execution of $\sigma_j = (r_j, q_j, d_j)$, with $i \neq j$. If we consider the particular case where $r_i = r_j$ and $t_i^1 = t_j^1$ (our model allows such a situation), then we have $\sigma_i^1 = [r_i, t_i^1) = [r_j, t_j^1) = \sigma_j^1$. Of course, in this paper, we consider that the intervals are *distinct* (i.e. $\sigma_i^1 \neq \sigma_j^1$). That is why we should define an interval σ_i by (r_i, q_i, d_i, i) and an effective execution σ_i^1 by $([r_i, t_i^1), i)$. But, in order to simplify the notations, we write $\sigma_i = (r_i, q_i, d_i)$ instead of $\sigma_i = (r_i, q_i, d_i, i)$, and $\sigma_i^1 = [r_i, t_i^1)$ instead of $\sigma_i^1 = ([r_i, t_i^1), i)$.

The algorithm AB^k. The main idea is the following. AB^k is running on k identical machines (called *real* machines because it is on these machines that the effective schedule is built). It uses as subroutines GOL and LR (described in Section 1). For the ease of notation, we use A for GOL and B for LR. For each new submitted interval σ_i, we simulate the execution of the algorithm A^r (resp. B^{k-r}) on r (resp. $k-r$) *virtual* machines, in order to control the size (resp. the weight) of the schedule. These two simulations (for the size and for the weight) are made on machines that we call *virtual*, because they are used only in order to determine the set (potentially empty) of intervals AB^k has to interrupt and whether σ_i has to be rejected or served by AB^k (and in this last case, to decide in which degraded version AB^k has to serve the new interval). Indeed, AB^k serves σ_i on a *real* machine if and only if A^r or B^{k-r} serves σ_i (note that if both A^r and B^{k-r} serve it, AB^k chooses the effective execution of σ_i that covers the other).

In order to determine the schedule given by an algorithm after the interrupting and the scheduling stages, we introduce the following notation.

Notation 2 (Schedule returned by an algorithm on step i) *For every on-line sequence $\sigma_1, \ldots, \sigma_n$, for every algorithm ALG and for every step of execution i ($1 \leq i \leq n$) of ALG, let $\mathcal{O}_{i_1}(ALG)$ (resp. $\mathcal{O}_{i_2}(ALG)$) be the schedule returned ALG after the* interrupting *(resp.* scheduling*) stage of step i.*

Notation 3 (Set of intervals scheduled by AB^k) *For every on-line sequence $\sigma_1, \ldots, \sigma_n$, for every step of execution i ($1 \leq i \leq n$) of the algorithm AB^k, let $\mathcal{R}_{i_1}(AB^k)$ (resp. $\mathcal{R}_{i_2}(AB^k)$) be the set of intervals scheduled and not interrupted after the* interrupting *(resp. the* scheduling*) stage of step i on the k machines associated to AB^k, called* real *machines.*

Notation 4 (Set of intervals scheduled by A^r and B^{k-r}) *For every on-line sequence $\sigma_1, \ldots, \sigma_n$, for every step of execution i ($1 \leq i \leq n$) 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 intervals scheduled and not interrupted after the* interrupting stage *of step i on the r (resp $k-r$) machines associated to A^r (resp. B^{k-r}). Let $\mathcal{V}_{i_2}(A^r)$ (resp. $\mathcal{V}_{i_2}(B^{k-r})$) be the set of intervals scheduled and not interrupted after the* scheduling stage *of step i on the r (resp. $k-r$) machines associated to A^r (resp. B^{k-r}). The r (resp $k-r$) machines associated to A^r (resp. B^{k-r}) are called* virtual *machines.*

We give now a formal description of the algorithm AB^k.

Input: An on-line sequence of intervals $\sigma_1, \ldots, \sigma_n$ and k identical machines.

Output: After each step i ($1 \leq i \leq n$), a valid schedule $\mathcal{O}_{i_2}(AB^k)$ of $\sigma_1, \ldots, \sigma_i$ on the 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) Execute the *interrupting stage* of AB^k on the k real machines associated to AB^k by interrupting the subset of intervals of $\mathcal{R}_{(i-1)_2}(AB^k)$ such that: $\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})$

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) Execute the *scheduling stage* of AB^k on the k real machines associated to AB^k by switching to the appropriate case:

i. If A^r and B^{k-r} reject σ_i, then AB^k does not schedule σ_i. Thus, we have: $\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k)$

ii. If A^r serves σ_i (with effective execution σ_i^A) and B^{k-r} rejects σ_i then AB^k serves σ_i^A on any free machine and we have:
$$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^A\}$$

iii. If A^r rejects σ_i and B^{k-r} serves σ_i (with effective execution σ_i^B) then AB^k serves σ_i^B on any free machine and we have:
$$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^B\}$$

iv. If A^r and B^{k-r} serve σ_i one with effective execution σ_i^A and the other with effective execution σ_i^B then AB^k serves the effective execution that covers the other on any free machine. If σ_i^B covers σ_i^A then we have:
$$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^B\}$$
else we have:
$$\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^A\}$$

Intervals scheduled by the algorithm AB^k. We first present Lemma 3 which states that the algorithm AB^k schedules the same intervals as the union$^+$ of the intervals scheduled by A^r and the intervals scheduled by B^{k-r}.

Lemma 3 *For each step i of execution of the algorithm AB^k, the schedule $\mathcal{O}_{i_2}(AB^k)$ is valid and $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$.*

Proof. We prove Lemma 3 by induction on the steps of execution i of AB^k.
The basic case (step 0): By definition of AB^k, we have $\mathcal{V}_{0_2}(A^r) = \mathcal{V}_{0_2}(B^{k-r}) = \mathcal{R}_{0_2}(AB^k) = \emptyset$. Thus, $\mathcal{O}_{i_2}(AB^k)$ is valid and we have $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$. The basic case is checked.
The main case (step i): Let us assume that $\mathcal{O}_{(i-1)_2}(AB^k)$ is valid and that $\mathcal{R}_{(i-1)_2}(AB^k) = \mathcal{V}_{(i-1)_2}(A^r) \uplus \mathcal{V}_{(i-1)_2}(B^{k-r})$ (by the assumption of the induction).

1. The interrupting stage: We first need to prove that $\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})$ and that $\mathcal{O}_{i_1}(AB^k)$ is valid.

(a) By definition, AB^k interrupts the subset of intervals of $\mathcal{R}_{(i-1)_2}(AB^k)$ such that:
$$\mathcal{R}_{i_1}(AB^k) = \mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r}) \qquad \text{(UNION)}$$

We have to show that there is always a subset of intervals 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 $\mathcal{R}_{(i-1)_2}(AB^k) = \mathcal{V}_{(i-1)_2}(A^r) \uplus \mathcal{V}_{(i-1)_2}(B^{k-r})$ (by the assumption of the induction), we have $\mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r}) \subseteq \mathcal{R}_{(i-1)_2}(AB^k)$.

(b) By definition, AB^k interrupts only intervals scheduled in $\mathcal{O}_{(i-1)_2}(AB^k)$, and by assumption of induction, $\mathcal{O}_{(i-1)_2}(AB^k)$ is valid. Thus, there cannot be intervals scheduled at the same time or more than once in $\mathcal{O}_{i_1}(AB^k)$. This means that $\mathcal{O}_{i_1}(AB^k)$ is valid. (VALID)

2. The scheduling stage: We now prove that $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$ and that $\mathcal{O}_{i_2}(AB^k)$ is valid. By definition of AB^k, several cases may happen:
   (a) If A^r and B^{k-r} reject σ_i, then AB^k does not schedule σ_i and we have:
      i.
$$\begin{aligned}\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \;&=\; \mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})\\ &\quad\text{(by the definition of } AB^k \text{ and by (UNION))}\\ &= \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})\\ &\quad\text{(since } A^r \text{ and } B^{k-r} \text{ reject } \sigma_i\text{, we have}\\ &\quad \mathcal{V}_{i_1}(A^r) = \mathcal{V}_{i_2}(A^r) \text{ and } \mathcal{V}_{i_1}(B^{k-r}) = \mathcal{V}_{i_2}(B^{k-r}))\end{aligned}$$
      ii. $\mathcal{O}_{i_2}(AB^k) = \mathcal{O}_{i_1}(AB^k)$. Thus $\mathcal{O}_{i_2}(AB^k)$ is valid (because by (VALID), $\mathcal{O}_{i_1}(AB^k)$ is valid).
   (b) If A^r serves σ_i (with effective execution σ_i^A) and B^{k-r} rejects σ_i, then AB^k schedules σ_i^A on any free real machine at time r_i and we have:
      i.
$$\begin{aligned}\mathcal{R}_{i_2}(AB^k) = \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^A\} \;&=\; (\mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})) \cup \{\sigma_i^A\}\\ &\quad\text{(by the definition of } AB^k \text{ and by (UNION))}\\ &= \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})\\ &\quad\text{(union and union}^+\text{ commute and since } A^r \text{ serves } \sigma_i\\ &\quad\text{and } B^{k-r}\text{rejects } \sigma_i\text{, we have } \mathcal{V}_{i_2}(A^r) = \mathcal{V}_{i_1}(A^r) \cup \{\sigma_i^A\}\\ &\quad\text{and } \mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_1}(B^{k-r}))\end{aligned}$$
      ii. Since $\mathcal{O}_{i_1}(AB^k)$ is a valid schedule (by (VALID)) and $\mathcal{O}_{i_2}(AB^k)$ is built by AB^k 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 are 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 $\mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$ 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 just proved above that $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$). Thus, $\mathcal{O}_{i_2}(AB^k)$ is a valid schedule.
   (c) If A^r rejects σ_i and B^{k-r} serves σ_i (with effective execution σ_i^B), then AB^k schedules σ_i^B on any free real machine at time r_i.

i. We prove that $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r})$ in the same way that we prove it in 2(b)i, except that we replace σ_i^A by σ_i^B.

ii. We prove that $\mathcal{O}_{i_2}(AB^k)$ is valid in the same way as in 2(b)ii.

(d) If A^r and B^{k-r} serve σ_i one with effective execution σ_i^A and the other with effective execution σ_i^B. Let σ_i^S (resp. σ_i^L) be the shortest (resp. longest) effective execution of σ_i. Without loss of generality, we suppose that A^r schedules σ_i^S and B^{k-r} schedules σ_i^L. By definition of the algorithm, AB^k schedules σ_i^L, and we have:

i.

$$
\begin{aligned}
\mathcal{R}_{i_2}(AB^k) &= \mathcal{R}_{i_1}(AB^k) \cup \{\sigma_i^L\} \;=\; (\mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})) \cup \{\sigma_i^L\} \\
&\quad \text{(by definition of } AB^k \text{ and by (UNION))} \\
&= (\mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})) \cup (\{\sigma_i^L\} \uplus \{\sigma_i^S\}) \\
&\quad \text{(because, by definition of union}^+\text{, } \{\sigma_i^L\} = \{\sigma_i^L\} \uplus \{\sigma_i^S\}) \\
&= \mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r}) \uplus \{\sigma_i^L\} \uplus \{\sigma_i^S\} \\
&\quad \text{(because } (\mathcal{V}_{i_1}(A^r) \uplus \mathcal{V}_{i_1}(B^{k-r})) \cap (\{\sigma_i^L\} \uplus \{\sigma_i^S\}) = \emptyset) \\
&= (\mathcal{V}_{i_1}(A^r) \cup \{\sigma_i^S\}) \uplus (\mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i^L\}) \\
&\quad \text{(because the union}^+ \text{ is commutative and since } \sigma_i^S \notin \\
&\quad \mathcal{V}_{i_1}(A^r)\text{, we have } \mathcal{V}_{i_1}(A^r) \uplus \{\sigma_i^S\} = \mathcal{V}_{i_1}(A^r) \cup \{\sigma_i^S\}) \\
&= \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r}) \\
&\quad \text{(because since } A^r \text{ and } B^{k-r} \text{ serve } \sigma_i\text{, we have } \mathcal{V}_{i_2}(A^r) \\
&\quad = \mathcal{V}_{i_1}(A^r) \cup \{\sigma_i^S\} \text{ and } \mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_1}(B^{k-r}) \cup \{\sigma_i^L\})
\end{aligned}
$$

ii. We prove that $\mathcal{O}_{i_2}(AB^k)$ is valid in the same way as in 2(b)ii.

□

Corollary 2 *Let $A^r = GOL^r$ and $B^{k-r} = LR^{k-r}$. Let $N(\mathcal{V}_{i_2}(GOL^r)) = |\mathcal{V}_{i_2}(GOL^r)|$ and $W(\mathcal{V}_{i_2}(LR^{k-r}))$ be the sum of the weight of the intervals of $\mathcal{V}_{i_2}(LR^{k-r})$. For every input sequence $\sigma_1, \ldots, \sigma_n$ and for every step i $(1 \le i \le n)$ of the algorithm AB^k, we have:*

$$N(\mathcal{V}_{i_2}(GOL^r)) \le N(\mathcal{R}_{i_2}(AB^k)) \quad \text{and} \quad W(\mathcal{V}_{i_2}(LR^{k-r})) \le W(\mathcal{R}_{i_2}(AB^k))$$

Proof. By Lemma 3, for every step i of the algorithm AB^k, we have $\mathcal{R}_{i_2}(AB^k) = \mathcal{V}_{i_2}(A^r) \uplus \mathcal{V}_{i_2}(B^{k-r}) = \mathcal{V}_{i_2}(GOL^r) \uplus \mathcal{V}_{i_2}(LR^{k-r})$, thus, by definition of union$^+$, Corollary 2 is checked. □

Theorem 1. *For all $k \ge 4$, for all r, $1 \le r \le k-3$, the algorithm AB^k applied with GOL^r and LR^{k-r} is $\left(\frac{k}{r}, \frac{4k}{k-r-2}\right)$-competitive for the Size and Proportional weights metrics.*

Proof. Let $\sigma_1, \cdots, \sigma_n$ be any on-line sequence of intervals and let O_x^{N*} (resp. O_x^{W*}) be an optimal schedule of $\{\sigma_1, \cdots, \sigma_n\}$ for the size N (resp. for the proportional weight W) on $x \le k$ machines. Let O_r^{GOL} (resp. O_{k-r}^{LR}) be the schedule returned by GOL^r (resp. LR^{k-r}) on the on-line sequence $\sigma_1, \cdots, \sigma_n$ on $r \le k-3$ (resp. $k-r \ge 3$) machines. Since, by Lemma 1, GOL^r is 1-competitive (resp. by Corollary 1, LR^{k-r} is $\left(\frac{4}{1-\frac{2}{k-r}}\right)$-competitive), we have:

$$N(O_r^{N*}) \leq N(O_r^{GOL}) \quad \text{(resp. } W(O_{k-r}^{W*}) \leq \left(\frac{4}{1-\frac{2}{k-r}}\right) W(O_{k-r}^{LR}) \text{)} \tag{1}$$

Let O'^N (resp. O'^W) be the r (resp. $k-r$) machine sub-schedule of O_k^{N*} (resp. O_k^{W*}) executing all the intervals appearing on the r (resp. $k-r$) machines of O_k^{N*} (resp. O_k^{W*}) generating the largest size (resp. weight). Since O'^N (resp. O'^W) is a r (resp. $k-r$) machine schedule, we have $N(O'^N) \leq N(O_r^{N*})$ (resp. $W(O'^W) \leq W(O_{k-r}^{W*})$), otherwise, O_r^{N*} (resp. O_{k-r}^{W*}) would not be an optimal schedule for the *size* (resp. *weight*). Combined with (1), we obtain:

$$\begin{array}{c} N(O'^N) \leq N(O_r^{N*}) \leq N(O_r^{GOL}) \\ \text{(resp. } W(O'^W) \leq W(O_{k-r}^{W*}) \leq \left(\frac{4}{1-\frac{2}{k-r}}\right) W(O_{k-r}^{LR}) \text{)} \end{array} \tag{2}$$

Since O'^N (resp. O'^W) is the r machine sub-schedule of O_k^{N*} (resp. O_k^{W*}) generating the largest size (resp. weight), the average size (resp. weight) per machine in O'^N (resp. O'^W) is larger than the average size (resp. weight) per machine in O_k^{N*} (resp. O_k^{W*}). Thus, we have $\frac{N(O_k^{N*})}{k} \leq \frac{N(O'^N)}{r} \Rightarrow N(O_k^{N*}) \leq \frac{k}{r} N(O'^N)$ (resp. $\frac{W(O_k^{W*})}{k} \leq \frac{W(O'^W)}{k-r} \Rightarrow W(O_k^{W*}) \leq \frac{k}{k-r} W(O'^W)$). Combined with (2), we obtain:

$$N(O_k^{N*}) \leq \frac{k}{r} N(O_r^{GOL}) \quad \text{(resp. } W(O_k^{W*}) \leq \frac{4k}{k-r-2} W(O_{k-r}^{LR})\text{)} \tag{3}$$

As $N(O_r^{GOL}) = N(\mathcal{V}_{i_2}(GOL^r))$ (resp. $W(O_{k-r}^{LR}) = W(\mathcal{V}_{i_2}(LR^{k-r}))$), by applying Corollary 2 on (3), we obtain:

$$\begin{array}{c} N(O_k^{N*}) \leq \frac{k}{r} N(\mathcal{V}_{i_2}(GOL^r)) \leq \frac{k}{r} N(\mathcal{R}_{i_2}(AB^k)) \\ \text{(resp. } W(O_k^{W*}) \leq \frac{k}{r} W(\mathcal{V}_{i_2}(LR^{k-r})) \leq \frac{4k}{k-r-2} W(\mathcal{R}_{i_2}(AB^k)) \text{)} \end{array}$$

This means that AB^k is $(\frac{k}{r}, \frac{4k}{k-r-2})$-competitive. □

Example. If $r = \frac{k}{2}$ (if $\frac{k}{2} \geq 3$ and k even), AB^k is $\left(2, \frac{8}{1-\frac{4}{k}}\right)$-competitive.

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, *Maximization of the size and the weight of schedules of degradable intervals*, in proceedings of COCOON'04, K.-Y. Chwa and I. Munro, eds., LNCS No. 3106, Springer-Verlag, 2004, pp. 219–228.
3. ———, *A note on bicriteria schedules with optimal approximation ratios*, Parallel Processing Letters, 14 (2004), pp. 315–323.
4. A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber, *Bandwidth allocation with preemption*, SIAM J. Comput., 28 (1999), pp. 1806–1828.
5. A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University press, 1998.
6. M. C. Carlisle and E. L. Lloyd, *On the k-coloring of intervals*, Discrete Applied Mathemetics, 59 (1995), pp. 225–235.
7. U. Faigle and M. Nawijn, *Note on scheduling intervals on-line*, Discrete Applied Mathematics, 58 (1995), pp. 13–17.
8. G. J. Woeginger, *On-line scheduling of jobs with fixed start and end times*, Theor. Comput. Sci., 130 (1994), pp. 5–16.