# On-line Time-Constrained Scheduling Problem for the Size on k machines

Nicolas Thibault, Christian Laforest

# On-line Time-Constrained Scheduling Problem for the Size on $k$ machines

Nicolas Thibault and Christian Laforest
Tour Evry 2, LaMI, Université d'Evry, 523 place des terrasses, 91000 EVRY France
{nthibaul, laforest}@lami.univ-evry.fr

## Abstract

*In this paper we consider the problem of scheduling on-line jobs on $k$ identical machines. Technically, our system is composed of $k$ identical machines and each* job *is defined by a triplet $\Gamma = (l, r, p)$, where $l$ denotes its left border, $r$ its right border and $p$ its length. When a job is revealed, it can be rejected or scheduled on one of the $k$ machines. In this last case, it can suppress already scheduled jobs. The goal is to maximize the* size *of the schedule (i.e. the* number of jobs *scheduled and not (later) suppressed). We propose an algorithm called $OLUW$. It is $(4 \min(\beta, \lfloor \log_2(\gamma) \rfloor + 1))$-competitive, where $\beta$ is the number of different job lengths appearing in the on-line input sequence and $\gamma$ is the ratio between the length of the longest job and the length of the shortest job in the sequence. To the best of our knowledge, $OLUW$ is the first on-line algorithm maximizing the size with guarantees on the competitive ratio for the time-constrained scheduling problem on $k$ identical machines.*

## 1 Introduction

We consider the problem of scheduling on-line *jobs* on $k$ identical machines (for any $k \geq 1$). We define a job $\Gamma$ by the triplet $\Gamma = (l, r, p)$, where $l$ is the *left border*, $r$ the *right border* and $p \leq r - l$ the *length* of $\Gamma$. A job is scheduled on machine $j$ if it occupies machine $j$ on a continuous interval of length $p$ between its two borders. Jobs are independent and two jobs scheduled on the same machine cannot intersect.

**Previous works.** This problem is known as the *Time-Constrained Scheduling Problem (TCSP)*. The aim is to maximize the weight of the constructed schedule (i.e. the sum of the weight of scheduled jobs). There are several categories of weights: the same for all jobs (corresponding to the maximization of the *size* of the schedule), equal to the length or arbitrary. In [5], all these variants of TCSP are studied in the off-line setting. As each variant is NP-hard, approximation algorithms are proposed, with constant approximation ratios. But the methods in [5] are off-line, thus they cannot be used for solving our on-line problem.

In the on-line setting (where jobs are revealed and treated one by one) particular variants of TCSP have been studied. The version where a job is an interval (i.e. with tight left and right borders, $p = r - l$) has been extensively studied. In [2, 3, 4, 7, 9, 12], algorithms for this model are proposed.

More recently, several papers [1, 8, 10, 11] have investigated variant of on-line TCSP and have proposed algorithms for the case where jobs have not tight left and right borders. But all these papers only consider the particular case of a single machine.

In this paper, we propose the first on-line algorithm solving TCSP for the *unit weight* model (i.e. for the maximization of the size of the schedule) on $k$ identical machines (in part, our work is more general than previous works on a single machine).

**Definitions.** We describe now more precisely our notations. When a job $\Gamma = (l, r, p)$ is revealed, all informations in the triplet $(l, r, p)$ are revealed. Job $\Gamma$ is said to be *scheduled* on machine $j$ if it continuously occupies machine $j$ between $l_0$ and $r_0$ with: $p = r_0 - l_0$ and $l \leq l_0 \leq r_0 \leq r$. The resulting numerical interval $\sigma = [l_0, r_0)$ is called the *interval associated* to $\Gamma$. Note that the length of $\sigma$ is $p$ and that $\sigma \subseteq [l, r)$ (i.e. job $\Gamma$ is scheduled between its two borders on a length $p$). A *schedule* $S$ is *feasible* if on any machine, no scheduled jobs (i.e. their associated intervals) intersect. The *size* $|S|$ of $S$ is the number of scheduled jobs. In the following, we will denote by $S_j$ the sub-part of a schedule $S$ on machine $j$.

Given any on-line sequence of jobs $\Gamma_1, \ldots, \Gamma_n, \ldots$ revealed one by one in this order, any on-line algorithm must construct at each step a *feasible* schedule on $k$ machines. In our model, when a new job $\Gamma = (l, r, p)$ is revealed, an on-line algorithm can *reject* it or *schedule* it. In this second case, if it schedules $\Gamma$ as the interval $\sigma$ on machine number $j$, it *interrupts* all the already scheduled intervals intersecting $\sigma$ on this machine (rejected jobs and interrupted intervals are definitively lost and do not appear in the current and future schedules). In order to evaluate the quality of an on-line scheduling algorithm, we introduce the following definition of the competitive ratio (see [6] for references on competitive ratios).

**Definition 1 (Competitive ratio)** *Let $\Gamma_1, \dots, \Gamma_n, \dots$ be any on-line sequence of jobs revealed in this order to on-line Algorithm $A$. Let $S^n$ be the corresponding schedule on $k$ machines constructed by Algorithm $A$ at step $n \geq 1$. Let $S^{n*}$ be the (optimal off-line) maximum size schedule on $k$ machines of the set of jobs $\{\Gamma_1, \dots, \Gamma_n\}$. Algorithm $A$ has a competitive ratio of $c$ (or is $c$-competitive) if and only if:*

$$\forall n \geq 1,\ c \cdot |S^n| \geq |S^{n*}|$$

**Application to networks.** Our model can be applied (but is not limited) to the following situation. Consider a link in a communication network, made of $k$ sub-links of identical capacities (for example an optical fiber in which $k$ independent frequencies can be used simultaneously). To schedule on-line requests (revealed one by one) on this link, an on-line algorithm has to choose which one is accepted and on which sub-link. In this application, one request is defined by a length (corresponding to the duration of transmission of the request on one sub-link), a release date and a deadline (after which completing the request is of no use). Here, we want to maximize the number of accepted requests.

**Outline of the paper.** In Section 2, we propose an on-line algorithm (called $OLUW$), solving TCSP on $k$ identical machines. In Section 3, we prove it is $(4\min(\beta, \lfloor\log_2(\gamma)\rfloor + 1))$-competitive, where $\beta$ is the number of different job lengths appearing in the on-line input sequence and $\gamma$ is the ratio between the length of the longest job and the length of the shortest job in the input sequence. In the particular case of a single machine system (i.e. $k = 1$), our algorithm has (in order of magnitude) the best possible competitive ratio, since it is proved in [9] that even for the interval model, no on-line algorithm has a competitive ratio better than $\Omega(\log\gamma)$.

## 2 Our algorithm $OLUW$

We define Algorithm $OLUW$ as follows.

*On-line Unit Weight Algorithm - $OLUW$*
Let $S$ be the current schedule made
of $k$ sub-schedules $S_j$ $(1 \leq j \leq k)$,
one on each machine number $j$ and
let $\Gamma = (l, r, p)$ be the new revealed job.
    IF there exists a machine number $j$
    such that there exists an interval
    $\sigma \subseteq [l, r)$ of length $p$ satisfying:
        $(\forall\sigma' \in S_j,\ \sigma \cap \sigma' = \emptyset)$ or
        $(\exists\sigma_0 \in S_j,\ \sigma \subset \sigma_0$ and $2p(\sigma) \leq p(\sigma_0))$
        THEN interrupt $\sigma_0$ (if necessary) and
        schedule $\Gamma$ as interval $\sigma$ on machine $j$
        ELSE reject $\Gamma$.

Note that at each step, Algorithm $OLUW$ constructs a feasible schedule.

To simplify the notations, for every interval $\sigma$, we say that $\sigma$ satisfies cond$_j(\sigma)$ if and only if $(\forall\sigma' \in S_j,\ \sigma \cap \sigma' = \emptyset)$ or $(\exists\sigma_0 \in S_j,\ \sigma \subset \sigma_0$ and $2p(\sigma) \leq p(\sigma_0))$. By using the following algorithm (called $IB$ for Important Borders, running on one machine), finding an interval $\sigma \subseteq [l, r)$ of length $p$ satisfying cond$_j(\sigma)$ if there exists one (corresponding to lines 4 and 5 of Algorithm $OLUW$) can be done in polynomial time for each machine $j$.

*Important Borders Algorithm - $IB$*
Let $\Gamma = (l, r, p)$ be the new revealed job.
Let $j$, $1 \leq j \leq k$ be the number of the machine
currently checked. On machine number $j$:
    Let $\{[l_1, r_1), \dots, [l_n, r_n)\}$ be the
    set of intervals already scheduled
    on machine number $j$ such that
    $l < r_1 \leq l_2 < r_2 \leq \dots \leq l_n < r_n < r$.
    Let $l, l_1, r_1, l_2, r_2, \dots, l_n, r_n$ be
    the sequence of *important borders*.
    Let $d$ be the first important border
    satisfying $[d, d + p) \subseteq [l, r)$
    and cond$_j([d, d + p))$.
    IF such a $d$ exists
        THEN there exists an interval
        $\sigma = [d, d + p) \subseteq [l, r)$ satisfying cond$_j(\sigma)$
        ELSE there is no interval $\sigma \subseteq [l, r)$
        of length $p$ satisfying cond$_j(\sigma)$.

The following Theorem proves the correctness of Algorithm $IB$.

**Theorem 1** *For every machine number $j$ ($1 \leq j \leq k$), Algorithm $IB$ finds an interval $\sigma = [d, d + p) \subseteq [l, r)$ of length $p$ satisfying cond$_j([d, d+p))$ if and only if there exists one.*

**Proof.** Of course, if Algorithm $IB$ finds an interval $\sigma = [d, d + p) \subseteq [l, r)$ of length $p$ satisfying cond$_j([d, d + p))$, then there exists one.

We now prove that if there exists an interval $\sigma \subseteq [l, r)$ of length $p$ satisfying cond$_j(\sigma)$, then Algorithm $IB$ finds one. Indeed, suppose, by contradiction, that there exists an interval $[l_0, r_0) \subseteq [l, r)$ of length $p$ (i.e. with $r_0 - l_0 = p$) satisfying cond$_j([l_0, r_0))$ and that Algorithm $IB$ finds no interval. Let $d_0 \in \{l, l_1, r_1, l_2, r_2, \dots, l_n, r_n\}$ be the largest important border such that $d_0 \leq l_0$. By definition of Algorithm $IB$, the interval $[d_0, d_0 + p)$ is checked. Let us now consider the two following cases:

- If $[l_0, r_0)$ intersects no interval scheduled on machine $j$. As $d_0$ is the largest important border such that $d_0 \leq l_0$ and as $[l_0, r_0)$ and $[d_0, d_0 + p)$ both have length $p$, $[d_0, d_0 + p)$ intersects no interval scheduled on machine $j$. This means that $[d_0, d_0 + p)$ satisfies $\mathrm{cond}_j([d_0, d_0 + p))$. Thus, Algorithm $IB$ chooses this interval (contradiction).

- Otherwise, there exists an interval $\sigma$ scheduled on machine $j$ intersecting $[l_0, r_0)$. As $[l_0, r_0)$ satisfies $\mathrm{cond}_j([l_0, r_0))$, we have $[l_0, r_0) \subseteq \sigma$. Moreover, as $d_0$ is the largest important border such that $d_0 \leq l_0$ and as $[l_0, r_0)$ and $[d_0, d_0 + p)$ both have length $p$, we also have $[d_0, d_0+p) \subseteq \sigma$. This means that $[d_0, d_0+p)$ satisfies $\mathrm{cond}_j([d_0, d_0+p))$. Thus, Algorithm $IB$ chooses this interval (contradiction).

□

## 3 Analysis of Algorithm $OLUW$

In the following, we consider any on-line sequence $\Gamma_1, \ldots, \Gamma_n, \ldots$ as input. In order to analyze its competitive ratio, we will compare $S$ (the schedule given by Algorithm $OLUW$ on $k$ machines at step $n$ for the input sequence $\Gamma_1, \ldots, \Gamma_n$) with $S^*$ (the optimal schedule given on $k$ machines at step $n$ for the input set $\{\Gamma_1, \ldots, \Gamma_n\}$).

**Main result.** In order to express our main result, we need the following definitions of the parameters $\beta$ and $\gamma$.

**Definition 2** *For every on-line sequence $\Gamma_1, \ldots, \Gamma_n$, we define $\beta$ as the number of different job lengths appearing in $\Gamma_1, \ldots, \Gamma_n$:*

$$\beta = \left| \bigcup_{\Gamma \in \Gamma_1, \ldots, \Gamma_n} \{p(\Gamma)\} \right|$$

**Definition 3** *For every on-line sequence $\Gamma_1, \ldots, \Gamma_n$, we define $\gamma$ as the ratio between the length of the longest job and the length of the shortest job in $\Gamma_1, \ldots, \Gamma_n$:*

$$\gamma = \max \left\{ \frac{p(\Gamma)}{p(\Gamma')} : \Gamma, \Gamma' \in \{\Gamma_1, \ldots, \Gamma_n\} \right\}$$

Our main result is the following theorem, expressed with the previous notations, proving that Algorithm $OLUW$ is $(4 \min(\beta, \lfloor \log_2(\gamma) \rfloor + 1))$-competitive.

**Theorem 2** $4 \min(\beta, \lfloor \log_2(\gamma) \rfloor + 1) \cdot |S| \geq |S^*|$

In order to prove Theorem 2, we need the following definitions.

**Notation 1** *For every $j$, ($1 \leq j \leq k$), let $S_j$ (resp. $S_j^*$) be the sub-schedule of $S$ (resp. $S^*$) on machine number $j$.*

**Notation 2** *For every machine number $j$ ($1 \leq j \leq k$), let $T_j$ be the set of associated intervals scheduled by Algorithm $OLUW$ on machine number $j$, including the intervals that have been scheduled and later interrupted.*

Note that two intervals in $T_j$ can overlap, since in general, $T_j$ is not a feasible schedule.

**Definition 4** *For every machine number $j$ ($1 \leq j \leq k$), we define $S_j^{*A}$ and $S_j^{*B}$ as follows:*

- *Let $S_j^{*A}$ be the sub-schedule of $S_j^*$ such that $S_j^{*A} \subseteq S_j^*$ and for every $\sigma_a \in S_j^{*A}$, there exists $\sigma_b \in \bigcup_{1 \leq j \leq k} T_j$ such that $\sigma_a$ and $\sigma_b$ are associated to the same job $\Gamma$ (i.e. $\Gamma$ is scheduled in $S_j^{*A}$ as interval $\sigma_a$ and has been accepted and scheduled by Algorithm $OLUW$ as interval $\sigma_b$).*

- *Let $S_j^{*B}$ be the sub-schedule of $S_j^*$ such that $S_j^{*B} \subseteq S_j^*$ and for every $\sigma_a \in S_j^{*B}$, there is no $\sigma_b \in \bigcup_{1 \leq j \leq k} T_j$ such that $\sigma_a$ and $\sigma_b$ are associated to the same job $\Gamma$ (i.e. $\Gamma$ is scheduled in $S_j^{*B}$ as interval $\sigma_a$ but has been rejected by Algorithm $OLUW$).*

Note that we have $S_j^{*A} \cap S_j^{*B} = \emptyset$ and $S_j^{*A} \cup S_j^{*B} = S_j^*$.

**Definition 5 (The function $f$)** *Let $j$ be any machine number ($1 \leq j \leq k$). Let $\Gamma_x$ be the new revealed job scheduled in the optimal solution as interval $x$ on machine $j$ and rejected by $OLUW$ (i.e. $x \in S_j^{*B}$). Let $S_j$ be the current sub-schedule of $S$ on machine $j$ when $\Gamma_x$ is revealed and let $T_j$ be the set of all associated intervals scheduled by Algorithm $OLUW$ on machine number $j$ before $\Gamma_x$ is revealed (including the intervals that have been scheduled and later interrupted). Note that both $S_j$ and $T_j$ are considered here as they currently are when job $\Gamma_x$ is revealed. Let $Y_x = \{y \in S_j : (p(y) < 2p(x)\ and\ x \subseteq y)\ or\ (x \cap y \neq \emptyset\ and\ x \not\subseteq y)\}$. We define the function $f$ as follows*

$$\begin{array}{rccl} f: & S_j^{*B} & \rightarrow & T_j \\ & x & \mapsto & y = [l_y, r_y) \text{ such that } y \in Y_x \\ & & & \text{and } l_y = \min\{l_{y'} : y' \in Y_x\} \end{array}$$

Note that we want $y$ be such that $l_y = \min\{l_{y'} : y' \in Y_x\}$ just to ensure that there is only one $y = f(x)$. We now introduce some technical Lemmas in order to prove Theorem 2.

**Lemma 1** *For every machine number $j$ ($1 \leq j \leq k$), for every interval $x \in S_j^{*B}$, we have*

$$|\{y : y = f(x)\}| = 1$$

**Proof.** Let $\Gamma_x$ be the new revealed job scheduled in $S_j^{*B}$ as interval $x$. Let $S_j$ be the current sub-schedule of $S$ on machine $j$ when $\Gamma_x$ is revealed. Since $x \in S_j^{*B}$, $\Gamma_x$ has been rejected by Algorithm $OLUW$. In particular, this means that Algorithm $OLUW$ has rejected $\Gamma_x$ of machine $j$. Thus, by definition of $OLUW$ and by Theorem 1, there exists $y_0 \in S_j$ such that $(p(y_0) < 2p(x)\ and\ x \subseteq y_0)\ or\ (x \cap y_0 \neq \emptyset\ and\ x \not\subseteq y_0)$. If there exist several such $y_0$, choose the one with the minimum left border. By definition of $f$, we have $y_0 = f(x)$, thus $|\{y : y = f(x)\}| = 1$. □

**Lemma 2** *Let $j$ be any machine number ($1 \leq j \leq k$). Let $\Gamma_x$ be the new revealed job scheduled in $S_j^{*B}$ as interval $x$. Let $S_j$ be the current sub-schedule of $S$ on machine $j$ when $\Gamma_x$ is revealed and let $T_j$ be the set of all associated intervals scheduled by Algorithm $OLUW$ on machine number $j$ before that $\Gamma_x$ is revealed (including the intervals that have been scheduled and later interrupted). For every interval $y \in T_j$, we have*

$$|\{x : y = f(x)\}| \leq 3$$

**Proof.** By contradiction, suppose that there exists $y \in T_j$ such that $|\{x : y = f(x)\}| \geq 4$. We denote by $\{x_1 = [l_1, r_1), x_2 = [l_2, r_2), \ldots, x_n = [l_n, r_n)\}$ (with $n \geq 4$) the set of intervals such that $\bigcup_{1 \leq l \leq n} \{x_l\} = \{x : y = f(x)\}$, ordered by increasing left borders (i.e. $l_1 < l_2 < \cdots < l_n$). As $y = f(x)$, we have

$$x \cap y \neq \emptyset \tag{1}$$

Moreover, since $x \in S_j^{*B} \subseteq S_j^*$, for every $a, b$ ($1 \leq a \leq b \leq n$), we have $x_a \cap x_b = \emptyset$ (because $S_j^*$ is a valid schedule on one machine). Thus, we have

$$r_1 \leq l_2 < r_2 \leq l_3 < \cdots < r_{n-1} \leq l_n \tag{2}$$

By (1) and (2), we obtain

$$\forall l,\ 2 \leq l \leq n-1,\ x_l \subset y \tag{3}$$

Without loss of generality, suppose that $p(x_2) = \min\left\{p(x) : x \in \bigcup_{2 \leq l \leq n-1} \{x_l\}\right\}$. Thus, we have

$$\begin{array}{rrcl} & (n-2)p(x_2) & \leq & \sum_{l=2}^{n-1} p(x_l) \\ \Rightarrow & 2p(x_2) & \leq & \sum_{l=2}^{n-1} p(x_l) \\ & & & \text{(because } n \geq 4) \\ \Rightarrow & 2p(x_2) & \leq & p(y) \\ & & & \text{(by (3))} \end{array}$$

This contradicts the fact that $y = f(x_2)$ (indeed, by definition of $f$, we have $p(y) < 2p(x_2)$). $\square$

**Lemma 3** $|S^*| \leq 4 \sum_{j=1}^{k} |T_j|$

**Proof.** By Lemma 1, for every machine $j$ ($1 \leq j \leq k$), we have $\{x : y \in T_j \ and \ y = f(x)\} = S_j^{*B}$. Thus, we have

$$|\{x : y \in T_j \ and \ y = f(x)\}| \ = \ |S_j^{*B}| \tag{4}$$

By Lemma 2, we have

$$|\{x : y \in T_j \ and \ y = f(x)\}| \ \leq \ 3|T_j| \tag{5}$$

By (4) and (5), we obtain

$$|S_j^{*B}| \ \leq \ 3|T_j| \tag{6}$$

Two cases may happen:

- If $|S^*| \leq 4 \sum_{j=1}^{k} |S_j^{*A}|$. By definition of $S_j^{*A}$, $\forall \sigma_a \in S_j^{*A}$, $\exists \sigma_b \in \bigcup_{1 \leq j \leq k} T_j$ such that $\sigma_a$ and $\sigma_b$ come from the same job $\Gamma$. Thus, we have $\sum_{j=1}^{k} |S_j^{*A}| \leq \sum_{j=1}^{k} |T_j|$. We obtain:

$$|S^*| \leq 4 \sum_{j=1}^{k} |T_j|$$

- If $|S^*| \geq 4 \sum_{j=1}^{k} |S_j^{*A}|$. By definition of $S_j^{*A}$ and $S_j^{*B}$, we have $S_j^{*A} \cap S_j^{*B} = \emptyset$ and $S_j^{*A} \cup S_j^{*B} = S_j^*$. Thus, we obtain:

$$\begin{array}{rrcl} & |S^*| & = & \sum_{j=1}^{k} |S_j^{*A}| + \sum_{j=1}^{k} |S_j^{*B}| \\ \Rightarrow & |S^*| & \leq & \frac{|S^*|}{4} + \sum_{j=1}^{k} |S_j^{*B}| \\ \Rightarrow & \frac{3}{4}|S^*| & \leq & \sum_{j=1}^{k} |S_j^{*B}| \\ \Rightarrow & |S^*| & \leq & \frac{4}{3} \sum_{j=1}^{k} |S_j^{*B}| \\ \Rightarrow & |S^*| & \leq & 4 \sum_{j=1}^{k} |T_j| \quad \text{(by (6))} \end{array}$$

$\square$

**Lemma 4** *For every machine number $j$ ($1 \leq j \leq k$), we have*

$$\min\left(\beta, \lfloor \log_2(\gamma) \rfloor + 1\right) \cdot |S_j| \geq |T_j|$$

**Proof.** For every $\sigma \in S_j$, we define the sequence of associated intervals "rooted" in $\sigma$ by $R(\sigma) = x_1, x_2, \ldots, x_i$ where $\sigma = x_i$ and $i$ is the largest integer such that for every $l$ ($1 \leq l \leq i-1$), $x_l$ has been replaced by $x_{l+1}$ during the execution of Algorithm $OLUW$. By definition of $OLUW$, we have

$$\begin{array}{rl} & p(x_1) \geq 2p(x_2) \geq 2^2 p(x_3) \geq \cdots \geq 2^{i-1} p(x_i) \\ \Rightarrow & \dfrac{p(x_1(\sigma))}{p(x_i(\sigma))} \ \geq \ 2^{i-1} \\ \Rightarrow & \gamma \ \geq \ 2^{i-1} \end{array}$$

(because $\gamma = \max\left\{\frac{p(\Gamma)}{p(\Gamma')} : \Gamma, \Gamma' \in \Gamma_1, \ldots, \Gamma_n\right\}$ and $x_1$ and $x_i$ are intervals associated to jobs in $\{\Gamma_1, \ldots, \Gamma_n\}$)

$$\Rightarrow \quad \lfloor \log_2(\gamma) \rfloor \ \geq \ |R(\sigma)| - 1$$

(because $|R(\sigma)| = i$ is an integer)

Thus, we obtain

$$\lfloor \log_2(\gamma) \rfloor + 1 \geq |R(\sigma)| \tag{7}$$

By definition of $\beta$, we have

$$\begin{aligned}
\beta &= \left| \bigcup_{\Gamma \in \{\Gamma_1, \ldots, \Gamma_n\}} \{p(\Gamma)\} \right| \\
&\geq \left| \bigcup_{\sigma' \in R(\sigma)} \{p(\sigma')\} \right| \\
&= |R(\sigma)| \quad \text{(because, by definition of } OLUW, \\
&\qquad\qquad p(x_1) > p(x_2) > \cdots > p(x_i)\text{)}
\end{aligned}$$

Thus, we obtain

$$\beta \geq |R(\sigma)| \tag{8}$$

By definition of $S_j$, $T_j$ and $R(\sigma)$, we have:

$$\begin{aligned}
& \bigcup_{\sigma \in S_j} R(\sigma) \;=\; T_j \\
\Rightarrow \quad & \left| \bigcup_{\sigma \in S_j} R(\sigma) \right| \;=\; |T_j| \\
\Rightarrow \quad & \sum_{\sigma \in S_j} |R(\sigma)| \;\geq\; |T_j| \\
\Rightarrow \quad & \sum_{\sigma \in S_j} \min\left(\beta, \lfloor \log_2(\gamma) \rfloor + 1\right) \;\geq\; |T_j| \\
& \text{(by (7) and (8))} \\
\Rightarrow \quad & \min\left(\beta, \lfloor \log_2(\gamma) \rfloor + 1\right) \cdot |S_j| \;\geq\; |T_j|
\end{aligned}$$

□

We are now able to give a proof of Theorem 2.

**Proof of Theorem 2.** By Lemmas 3 and 4, we have

$$\begin{aligned}
|S^*| &\leq 4 \textstyle\sum_{j=1}^{k} |T_j| \\
&\leq 4 \textstyle\sum_{j=1}^{k} \min\left(\beta, \lfloor \log_2(\gamma) \rfloor + 1\right) \cdot |S_j| \\
&= 4 \min\left(\beta, \lfloor \log_2(\gamma) \rfloor + 1\right) \cdot |S|
\end{aligned}$$

□

### 3.1 Conclusion

We have proposed a $(4 \min(\beta, \lfloor \log_2(\gamma) \rfloor + 1))$-competitive on-line algorithm (called $OLUW$), solving TCSP on $k$ identical machines, where $\beta$ is the number of different job lengths appearing in the on-line input sequence and $\gamma$ is the ratio between the length of the longest job and the length of the shortest job in the input sequence (note that Algorithm $OLUW$ do not need to know neither $\beta$ nor $\gamma$ beforehand). We underline the fact that in many situations (corresponding to "homogeneous" jobs), this competitive ratio is constant. For example, when $\beta = c$ (with $c$ any constant), Algorithm $OLUW$ is $4c$-competitive. When $\gamma = 2^{c'}$ (with $c'$ any constant), Algorithm $OLUW$ is $(4c' + 4)$-competitive.

## References


[1] R. Adler and Y. Azar. Beating the logarithmic lower bound: Randomized preemptive disjoint paths and call control algorithms. *Journal of Scheduling*, 6(2):113–129, 2003.

[2] F. Baille, E. Bampis, C. Laforest, and N. Thibault. On-line bicriteria interval scheduling. In *Euro-Par*, 2005.

[3] F. Baille, E. Bampis, C. Laforest, and N. Thibault. On-line simultaneous maximization of the size and the weight for degradable intervals schedules. In *COCOON*, 2005.

[4] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber. Bandwidth allocation with preemption. *SIAM J. Comput.*, 28(5):1806–1828, 1999.

[5] A. Bar-Noy, G. S., N. J., and S. B. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing*, 31:331–352, 2001.

[6] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University press, 1998.

[7] U. Faigle and W. Nawijn. Note on scheduling intervals on-line. *Discrete Applied Mathematics*, 58(58):13–17, 1994.

[8] J. Garay, J. Naor, B. Yener, and P. Zhao. On-line admission control and packet scheduling with interleaving. In *Proceedings of INFOCOM*, pages 94–103, 2002.

[9] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23(1):180–194, 1997.

[10] M. H. Goldwasser. Patience is a virtue: the effect of slack on competitiveness for admission control. In *10th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia*, pages 396–405, 1999.

[11] J. S. Naor, A. Rosen, and G. Scalosub. Online time-constrained scheduling in linear networks. In *Proceedings of INFOCOM*, 2005.

[12] G. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoritical Computer Science*, 130(130):5–16, 1994.