On the minimization of late jobs in single machine scheduling with nested execution intervals

Cyril Briand [a,*], Samia Ourari [b], Brahim Bouzouia [b]

[a] *LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse, France*

[b] *CDTA, lotissement du 20 août 1956, Baba Hassen, Alger, Algeria*

Abstract

This paper considers the problem of scheduling n jobs on a single machine. A fixed processing time and an execution interval are associated with each job. Job execution intervals are assumed to be nested each inside the other, like Russian dolls. The objective is to minimize the number of late jobs. For this problem, denoted as $1|r_i, nested|\sum U_i$, a new dominance condition and an optimal $O(n^3)$ solving procedure are proposed.

Key words: single machine scheduling, nested execution intervals, dominance conditions.

1 INTRODUCTION

A single machine problem consists of a set V of n jobs to be scheduled on a single disjunctive resource. The processing time p_j of each job j is known. A start time s_j has to be found for each job j, while respecting its release date r_j ($s_j \geq r_j$) and its due date d_j ($s_j + p_j \leq d_j$). The interval $[r_j, d_j]$ defines the execution window of each job j.

In this paper, the considered criterion is the minimization of the number of late jobs. That problem is denoted as $1|r_i|\sum U_i$ in the literature (U_i is set to 1 if job i is late: $U_i \leftarrow s_i + p_i > d_i$). It is NP-hard in the strong sense [8].

* Corresponding author.

Email addresses: briand@laas.fr (Cyril Briand), sourari@cdta.dz (Samia Ourari), bbouzouia@cdta.fr (Brahim Bouzouia).

Nevertheless, several exact branch and bound solving procedures are available [2,5]. Those methods are able to solve to optimality problems having up to 200 jobs.

When some additional assumptions are made, $1|r_i|\sum U_i$ problems can become solvable in polynomial time. For instance, when release dates are equal, $1||\sum U_i$ problems are solved in $O(n\log(n))$ using Moore's well known algorithm [12]. In the case where release and due dates of jobs are ordered similarly (i.e. $r_i < r_j \Rightarrow d_i \leq d_j$), Kise, Ibaraki and Mine [9] proposed in 1978 a dynamic programming algorithm that solves the problem in $O(n^2)$. Under that same assumption, an $O(n\log(n))$ algorithm was later proposed by Lawler in 1982 [10]. Lawler [11] also described an $O(n\log(n))$ algorithm that works on preemptive nested problems $1|r_j, nested, pmtn|\sum U_j$, i.e. job execution windows are nested $r_i < r_j \Rightarrow d_i \geq d_j$ or $d_i > d_j \Rightarrow r_i \leq r_j$ and job preemption is allowed. Considering the case where processing times are equal, Carlier [3] proved in 1984 that the problem $1|p_j = p, r_j|\sum U_j$ can be solved in $O(n^3\log(n))$. More recently, considering the general preemptive problem $1|r_j, pmtn|\sum U_j$, Baptiste designed an algorithm which runs in $O(n^4)$ [1].

Looking for new polynomial algorithms for solving particular $1|r_i|\sum U_i$ problems can be helpful to design new efficient lower bounds for the general problem. For instance, in [5], an efficient lower bound is implemented which is based on the Kise et al. procedure (release and due dates are relaxed so that they become similarly ordered). In the same spirit, Baptiste et al. [2], inside their branch-and-bound procedure, used an efficient lower bound that is based on the $1|r_j, nested, pmtn|\sum U_j$ polynomial solving procedure.

Keeping that aim in mind, this paper focuses on $1|r_i, nested|\sum U_i$ problems. To the best of our knowledge, since only the $1|r_j, nested, pmtn|\sum U_j$ has been tackled in the literature, it does not exist yet any polynomial procedure able to solve to optimality this problem.

The paper is structured as follows. In the 1st section, a general dominance theorem is recalled. Using the previous theorem, the 2nd section proposes a new dominance theorem for the $1|r_i, nested|\sum U_i$ problem. In the 3rd section, a new necessary and sufficient condition of admissibility is given for single machine problem with nested execution intervals. Using the previous condition, efficient dominance conditions are presented in section 5. The last section proposes an optimal $O(n^3)$ procedure for solving $1|r_i, nested|\sum U_i$ problems.

2 A GENERAL DOMINANCE THEOREM FOR SINGLE MACHINE SEQUENCING [7]

In this section, the $\sum U_i$ criterion is temporarily forgotten and we take an interest in admissibility of job sequences. The theorem below gives a dominance condition that allows to eliminate job sequences which are dominated, regarding the admissibility objective. Before presenting the theorem, the notions of *top* and *pyramid* need to be defined.

Definition 1. *A job $t \in V$ is a top if there does not exist any other job $i \in V$ such that $r_i > r_t \wedge d_i < d_t$.*

The tops are indexed according to the ascending order of their release dates or, in case of equality, according to the ascending order of their due dates. When both their release dates and due dates are equal, the tops are indexed in an arbitrary order. Thus, if t_α and t_β are two tops then $\alpha < \beta$ if and only if $(r_{t_\alpha} \leq r_{t_\beta}) \wedge (d_{t_\alpha} \leq d_{t_\beta})$.

Definition 2. *Given a top t_α, a pyramid P_α related to t_α is the set of jobs $i \in V$ such that $r_i < r_{t_\alpha} \wedge d_i > d_{t_\alpha}$.*

Considering the previous definition, it must be noticed that a non-top job can belong to several pyramids.

The dominance theorem below [7] can now be stated. It aims at pruning the solution space associated with deterministic single machine scheduling problems, so that finding an admissible solution becomes less time expensive.

Theorem 1. *A dominant set of sequences can be constituted by the sequences such that:*

- *the tops are ordered according to the ascending order of their index;*
- *only the jobs belonging to the first pyramid can be located before the first top and they are ordered according to the ascending order of their release dates (in an arbitrary order in case of release date equality);*
- *only the jobs belonging to the last pyramid can be located after the last top and they are ordered according to the ascending order of their due dates (in an arbitrary order in case of release date equality);*
- *only the jobs belonging to the pyramids P_k or P_{k+1} can be located between two successive tops t_k and t_{k+1} so that:*
  - *the jobs belonging only to P_k but not to P_{k+1} are sequenced immediately after t_k according to the ascending order of their due dates (in an arbitrary order in case of equality),*
  - *then the jobs belonging to both P_k and P_{k+1} are sequenced in an arbitrary order,*
  - *and lastly are sequenced the jobs belonging only to P_{k+1} but not to P_k in the ascending order of their release dates (in an arbitrary order in case of*

equality).

In the previous theorem, it must be pointed out that the processing times p_j as well as the explicit values of r_j and d_j are not used. Only the relative order of the release and due dates is considered, hence the genericity of the result.

For illustration, let us focus on a problem having seven jobs with job release dates r_j and due dates d_j ordered so that $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 < (r_5 = r_7) < d_6 < d_5 < d_1 < d_7$. There are four tops: $t_1 = 2$, $t_2 = 4$, $t_3 = 5$ and $t_4 = 7$, which characterize four pyramids: $P_1 = \{1, 3, 6\}$, $P_2 = \{1, 6\}$, $P_3 = \{1\}$ and $P_4 = \emptyset$ respectively. Note that, in accordance with Definition 2, a top does not belong to the pyramid it characterizes. Whatever the real values of r_i and d_i are (provided they are compatible with the previous relative order), Theorem 1 characterizes twenty four dominant sequences (out of $7! = 5040$):

$6 \prec 1 \prec 3 \prec \mathbf{2} \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$1 \prec 3 \prec \mathbf{2} \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec \mathbf{7}$,
$1 \prec \mathbf{2} \prec 3 \prec 6 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec 3 \prec \mathbf{2} \prec 1 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec 1 \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec 6 \prec 1 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec 3 \prec \mathbf{2} \prec \mathbf{4} \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec \mathbf{4} \prec 6 \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec 6 \prec \mathbf{4} \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec 3 \prec \mathbf{2} \prec \mathbf{4} \prec \mathbf{5} \prec 1 \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec 1 \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec 6 \prec \mathbf{4} \prec \mathbf{5} \prec 1 \prec \mathbf{7}$,
$1 \prec 3 \prec \mathbf{2} \prec 6 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec 1 \prec \mathbf{2} \prec 3 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$1 \prec \mathbf{2} \prec 3 \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec 6 \prec 1 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec \mathbf{2} \prec 3 \prec 1 \prec \mathbf{4} \prec \mathbf{5} \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec 1 \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec 6 \prec \mathbf{4} \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$6 \prec \mathbf{2} \prec 3 \prec \mathbf{4} \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec \mathbf{4} \prec 6 \prec 1 \prec \mathbf{5} \prec \mathbf{7}$,
$3 \prec \mathbf{2} \prec 6 \prec \mathbf{4} \prec \mathbf{5} \prec 1 \prec \mathbf{7}$,
$6 \prec \mathbf{2} \prec 3 \prec \mathbf{4} \prec \mathbf{5} \prec 1 \prec \mathbf{7}$,
$\mathbf{2} \prec 3 \prec \mathbf{4} \prec 6 \prec \mathbf{5} \prec 1 \prec \mathbf{7}$.

3 DOMINANCE AND MASTER-PYRAMID SEQUENCE

In this section, the $\sum U_i$ is considered back. Searching optimal solution for $1|r_i|\sum U_i$ problem amounts to determine admissible sequence for the largest possible selection of jobs. Let E^* be that selection. The jobs of E^* are on time while others are late. The late jobs can be scheduled after the jobs of E^* in any order. So they do not need to be considered when searching an admissible sequence for on-time jobs and Theorem 1 can be applied only to the jobs belonging to E^*. There are $\sum_{k=1\ldots n} A_n^k$ possible different selections of jobs. Regarding now the $\sum U_i$ criterion, the following corollary is proven.

Corollary 1. *The union of all the dominant sequences that Theorem 1 characterizes for each possible selection of jobs is dominant for the $\sum U_i$ criterion.*

Proof. The proof is obvious since the union of all the sequences that Theorem 1 characterizes for any possible selection necessarily includes the dominant

sequences associated with E^*, hence an optimal solution. □

As already pointed out, the number of possible job selections is quite large. Nevertheless, as it is discuss in [13], it is not necessary to enumerate all the possible job selections to get the dominant sequences. Indeed they can be characterized with one or more *master-pyramid sequences*. The notion of master-pyramid sequence is somewhat close to the notion of master sequence that Dauzère-Pérès and Sevaux proposed in [5]. It is illustrated below for the special case where there is only one top, hence one single pyramid P, and job execution intervals are nested. In that case, for matter of simplification, Pyramid P will be said *perfect*.

Let P be a perfect pyramid. The $n+1$ jobs belonging to P are supposed to be numbered according to the nesting order, i.e. so that:

$$r_n \leq \cdots \leq r_1 \leq r_0 \leq d_0 \leq d_1 \leq \cdots \leq d_n$$

Job 0 is the unique top. Job n is called the *base* of P. The following theorem can now be stated.

Theorem 2. *P being a perfect pyramid having its $n+1$ jobs numbered according to the nesting order, any sequence of $k < n+1$ jobs compatible with the master-pyramid sequence σ_Δ:*

$$\sigma_\Delta = \{n, n-1, \ldots, 1, 0, 1, \ldots, n-1, n\}$$

is dominant regarding the $\sum U_j$ criterion.

Before giving the proof, the notion of compatible sequence need to be clarified. A sequence s is said compatible with the master-pyramid sequence σ_Δ if the order of the jobs in s does not contradict the partial order defined by σ_Δ. In other words, in the current case, a sequence is compatible if and only if the number order of the jobs decreases and then increases (but never increases or decreases twice). For instance, given a perfect pyramid P with 5 jobs, the sequence of 4 jobs $(3, 1, 2, 4)$ is compatible with $\sigma_\Delta = \{4, 3, 2, 1, 0, 1, 2, 3, 4\}$. At the opposite, the sequence $(3, 1, 4, 2)$ is not compatible with σ_Δ.

Proof. The proof of Theorem 2 is obvious. Indeed, one can easily check that, for any selection E^k of k jobs, the corresponding master-pyramid sequence σ_Δ^k is compatible with the master sequence σ_Δ holding all the n jobs. Indeed, σ_Δ^k is in the form $\alpha^k \prec t^k \prec \beta^k$ where:

- t^k is the job of E^k having the smallest number;
- α^k is the sequence of the jobs of E^k (t^k excepted) ordered according the descending order of their number ;

- β^k is the sequence of the jobs of E^k (t^k excepted) ordered according the ascending order of their number.

Therefore any sequence compatible with σ^k_Δ is necessarily compatible with σ_Δ. □

Theorem 2 allows a significant reduction of the size of the search space. Indeed, for a selection E^k of $k+1$ jobs, only 2^k sequences need to be explored for admissibility instead of $(k+1)!$. Nevertheless, that size remains obviously too large for allowing a complete enumeration of the dominant sequences. Later on, another dominance theorem is stated that allows to reduce the search space much more. It is based on a necessary and sufficient condition of admissibility that is presented in the next section.

4 A NEW CONDITION OF ADMISSIBILITY FOR THE NESTED CASE

In this section, the $\sum U_j$ criterion is again temporarily forgotten and we take an interest in finding admissible sequences. In other words, we try to find a sequence so that, if it exists, all the jobs are on time. As stated in [6] regarding the single machine problem, let us first recall that a job sequence σ is admissible iff:

$$p_i + \sum_{i \prec k \prec j} p_k + p_j \leq d_j - r_i \ ,\forall (i,j) \in \sigma \text{ such that } i \prec j \qquad (1)$$

That necessary and sufficient condition can be drastically simplified when job execution intervals are nested. In the sequel, for matter of notation simplification, given a sequence σ in the form $\alpha \prec 0 \prec \beta$, we set:

- $R^a_\alpha = r_a + p_a + \sum_{k \in \alpha, a \prec k} p_k$;
- $D^b_\beta = d_b - p_b - \sum_{k \in \beta, k \prec b} p_k$;
- $R^{\max}_\alpha = \max_{a \in \alpha} (R^a_\alpha)$;
- $D^{\min}_\beta = \min_{b \in \beta} (D^b_\beta)$;
- $\Delta^{\min}_{\alpha \prec 0 \prec \beta} = \min(d_0, D^{\min}_\beta) - \max(r_0, R^{\max}_\alpha)$.

Theorem 3. *P being a perfect pyramid of top 0, any compatible sequence in the form $\alpha \prec 0 \prec \beta$ is admissible iff $\Delta^{min}_{\alpha \prec 0 \prec \beta} \geq p_0$.*

Proof. We first prove that if $\Delta^{\min}_{\alpha \prec 0 \prec \beta} \geq p_0$ then $\alpha \prec 0 \prec \beta$ is admissible. From Equation 1, $\alpha \prec 0 \prec \beta$ is admissible iff $d_t - r_t \geq p_0$ and $D^b_\beta - R^a_\alpha \geq p_0$, $\forall a \in \alpha$ and $\forall b \in \beta$. It is obvious since $D^b_\beta - R^a_\alpha \geq D^{\min}_\beta - R^{\max}_\alpha$ by definition and $\min(d_0, D^{\min}_\beta) - \max(r_0, R^{\max}_\alpha) \geq p_0$ by assumption.

Now, we prove that if $\alpha \prec 0 \prec \beta$ is admissible then $D^{\min}_{\beta} - R^{\max}_{\alpha} \geq p_0$. It is also obvious since if $\alpha \prec 0 \prec \beta$ is admissible then, from Condition 1, $D^b_\beta - R^a_\alpha \geq p_0, \forall (a,b)$ and $d_0 - r_0 \geq p_0$. Of course, that inequality need to be verified in the worst case, i.e. $\min(d_0, D^b_\beta) - \max(r_t, R^a_\alpha)$. Hence $\Delta^{min}_{\alpha \prec 0 \prec \beta} \geq p_0$. □

The fact that Theorem 3 allows to numerically compare sequences each other is important. That property is used in the section below.

5 NEW DOMINANCE CONDITIONS FOR THE NESTED CASE

Still regarding the admissibility, the necessary and sufficient admissibility condition presented in Theorem 3 immediately involves the following dominance condition.

Corollary 2. *P being a perfect pyramid, job 0 being its top, and s_1 and s_2 being two compatible sequences in the form $\alpha_1 \prec 0 \prec \beta_1$ and $\alpha_2 \prec 0 \prec \beta_2$ respectively, if $\Delta^{\min}_{\alpha_1 \prec 0 \prec \beta_1} \geq \Delta^{\min}_{\alpha_2 \prec 0 \prec \beta_2}$ then s_1 dominates s_2 regarding the admissibility.*

Proof. By definition, stating that s_1 dominates s_2 requires to prove that, if s_2 is admissible, then s_1 is also admissible. Now we know that s_2 is admissible iff (cf. Theorem 3) $\Delta^{\min}_{\alpha_2 \prec 0 \prec \beta_2} \geq p_0$, therefore, under the assumption that $\Delta^{\min}_{\alpha_1 \prec 0 \prec \beta_1} \geq \Delta^{\min}_{\alpha_2 \prec 0 \prec \beta_2}$, we deduce $\Delta^{\min}_{\alpha_1 \prec 0 \prec \beta_1} \geq p_0$, so s_1 is necessarily admissible. □

Corollary 2 allows to formulate the problem of searching an admissible job sequence as an integer linear problem:

$$\max \; z = D^{\min} - R^{\max}$$

$$\text{s.t.} \quad \begin{cases} R^{\max} \geq r_i + p_0 + \sum_{j=1}^{i} p_j y_j^- \\ D^{\min} \leq d_i - p_0 - \sum_{j=1}^{i} p_j y_j^+ \\ y_i^- + y_i^+ = 1 \\ y_i^-, y_i^+ \in \{0,1\} \\ D^{\min}, R^{\max} \in \mathbb{Z} \\ \forall i = 0 \ldots n \end{cases}$$

In the above formulation, y_i^- (y_i^+ resp.) is set to 1 if the job i is sequenced at the left of the top (at the right of the top resp.). The constraint $y_i^- + y_i^+ = 1$ imposes that any job is sequenced either at the left or at the right of the top. If $z = D^{\min} - R^{\max}$ is maximized while respecting the constraints then, from Corollary 2, the obtained sequence dominates all the others. Moreover,

if $z^* \geq -p_0$ then, from Theorem 3, that sequence is also admissible. Indeed, $D^{\min} = D^{\min}_{\beta} - p_0$ and $R^{\max} = R^{\max}_{\alpha} + p_0$, so $z = D^{\min}_{\beta} - R^{\max}_{\alpha} - 2p_0$, hence $\Delta^{min}_{\alpha \prec 0 \prec \beta} \geq p_0$ is equivalent to $z^* \geq -p_0$. If $z^* < -p_0$ then it can be asserted that it does not exist any admissible sequence for the considered problem.

By setting $x_i^- = 1 - y_i^-$ and $x_i^+ = 1 - y_i^+$, the above formulation becomes:

$$\max \; z = D^{\min} - R^{\max}$$

$$\text{s.t.} \quad \begin{cases} R^{\max} \geq r_i + \sum_{j=0}^{i} p_j - \sum_{j=1}^{i} p_j x_j^- \\ D^{\min} \leq d_i - \sum_{j=0}^{i} p_j + \sum_{j=1}^{i} p_j x_j^+ \\ x_i^- + x_i^+ = 1 \\ x_i^-, x_i^+ \in \{0,1\} \\ D^{\min}, R^{\max} \in \mathbb{Z} \\ \forall i = 0 \ldots n \end{cases}$$

The constraint matrix of that optimization problem is quite particular. Indeed firstly, it is bi-triangular and secondly, the column coefficients are identical (they correspond to the processing times p_i). This is why an optimal solution can be easily found by iteratively setting one x_i variable to 1, in such a way that the increase of z is maximized each time.

If $x_i^- \leftarrow 1$ (i is sequenced at the right of the top) then $x_i^+ \leftarrow 0$. The current criterion value z therefore increases because $R^{\max}$ decreases so that its value becomes $\max(R^{\max} - p_i, \max_{k \leq i-1} R_k)$, with $R_k = r_k + \sum_{j=0}^{k} p_j - \sum_{j=1}^{k} p_j x_j^-$. Similarly, if $x_i^+ \leftarrow 1$ (i is sequenced at the left of the top) then $x_i^- \leftarrow 0$ and z increases because the new value of $D^{\min}$ is $\min(D^{\min} - p_i, \min_{k \leq i-1} D_k)$, with $D_k = d_k - \sum_{j=0}^{k} p_j + \sum_{j=1}^{k} p_j x_j^+$.

So, by searching at each iteration which job r can be sequenced at the right of the top, so that the value of $\max(R^{\max} - p_i, \max_{k \leq i-1} R_k)$ is minimal, one can compute in $\mathrm{O}(n)$ the maximal increase δr of z. Similarly, by searching which job l can be sequenced at the left of the top, so that the value $\min(D^{\min} - p_i, \min_{k \leq i-1} D_k)$ is maximized, one can compute in $\mathrm{O}(n)$ the maximal increase δd of z. The global maximal increase of z is therefore $\max(\delta r, \delta d)$.

The solving procedure which computes a single dominant sequence is detailed in Algorithm 1. It runs in $\mathrm{O}(n^2)$. Let us notice that in the special case where $\delta r = \delta d$, the algorithm sequences l at the left of the top if $d_0 - R[r] > D[l] - r_0$, r at the right of the top otherwise, in order to favor the admissibility.

For illustration, let us consider the problem with 5 jobs of Table 1. The table 2 describes, for each iteration, the relevant values of the algorithm parameters

Algorithm 1 An $O(n^2)$ algorithm for finding an admissible sequence

Function FindAdm $(r_i, p_i, d_i,\ i = 0 \dots n)$
$\varepsilon \leftarrow 1, 2, \dots n$;
$\alpha \leftarrow \emptyset$;
$\beta \leftarrow \emptyset$;
$R[i] \leftarrow r_i + \sum_{j=0}^{i} p_j,\ \forall i = 0 \dots n$;
$D[i] \leftarrow d_i - \sum_{j=0}^{i} p_j,\ \forall i = 0 \dots n$;
$R^{\max} \leftarrow \max_{i=0\dots n} R[i]$;
$D^{\min} \leftarrow \min_{i=0\dots n} D[i]$;
for i $\leftarrow 1$ to n **do**
  $r \leftarrow \operatorname{argmin}_{j\in\varepsilon} \max(R^{\max} - p_j, \max_{k\le j-1} R[k])$;
  $\delta r \leftarrow R^{\max} - \max(R^{\max} - p_r, \max_{k\le r-1} R[k])$;
  $l \leftarrow \operatorname{argmax}_{j\in\varepsilon} \min(D^{\min} + p_j, \min_{k\le j-1} D[k])$;
  $\delta d \leftarrow \min(D^{\min} + p_l, \min_{k\le l-1} D[k]) - D^{\min}$;
  if $(\delta r > \delta d)$ **OR**
  $(\delta r = \delta d$ **AND** $d_0 - R[r] \le D[l] - r_0)$ **then**
    $\beta \leftarrow \beta + \{r\}$;
    $\varepsilon \leftarrow \varepsilon - \{r\}$;
    $R[j] \leftarrow R[j] - p_r,\ \forall j = r \dots n$;
    $R^{\max} \leftarrow \max_{j=0\dots n} R[i]$;
  else
    $\alpha \leftarrow \alpha + \{l\}$;
    $\varepsilon \leftarrow \varepsilon - \{l\}$;
    $D[j] \leftarrow D[j] + p_l,\ \forall j = l \dots n$;
    $D^{\min} \leftarrow \min_{j=0\dots n} R[j]$;
  end if
end for
Return $\alpha \prec 0 \prec \beta$;

i	0	1	2	3	4
r_i	9	5	3	2	0
p_i	2	3	5	3	5
d_i	12	15	16	18	19

Table 1
An example with 5 jobs

and the decision made. Figure 1 describes the job sequence that is returned. That sequence is admissible since $D^{\min} - R^{\max} = 10 - 12 = -2 \ge -p_0$. We highlight that, at the first iteration, because $\delta r = \delta d = 5$, the algorithm decides to sequence Job 3 in α since $d_0 - R[2] = -1$ is greater than $D[2] - r_0 = -3$.

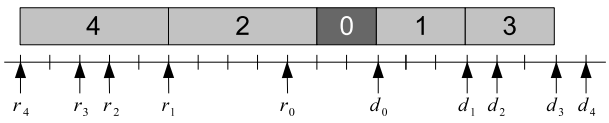

Fig. 1. The admissible sequence

i	1	2	3	4	5
R^{max}	18	18	18	15	12
D^{min}	1	6	10	10	10
l	2	1	1	3	-
δr	5	3	3	3	-
r	2	4	1	3	-
δd	5	4	0	0	-
$\alpha 0 \beta$	20	420	4201	42013	42013

Table 2
Algorithm parameters at each iteration i

6 AN OPTIMAL ALGORITHM FOR $1|r_i, nested|\sum U_i$ PROBLEMS

In this section, the $\sum U_i$ criterion is again considered. We first take an interest in the special problem of minimizing the number of late jobs (with nested execution intervals), under the strong assumption that the top job must be on time. That problem is quite similar to the admissibility problem discussed in section 5 since it can be formulated as follows:

$$\min\ w = \sum_{i=1}^{n} x_i^+ + x_i^-$$

$$\text{s.t.} \begin{cases} R^{\max} \geq r_i + \sum_{j=0}^{i} p_j - \sum_{j=1}^{i} p_j x_j^- \\ D^{\min} \leq d_i - \sum_{j=0}^{i} p_j + \sum_{j=1}^{i} p_j x_j^+ \\ x_i^- + x_i^+ \geq 1 \\ D^{\min} - R^{\max} \geq -p_0 \\ x_i^-, x_i^+ \in \{0,1\} \\ D^{\min}, R^{\max} \in \mathbb{Z} \\ \forall i = 0 \dots n \end{cases}$$

The constraints of that problem are the same than the ones of the admissibility problem excepting that:

- the solution has to be admissible (i.e. $D^{\min} - R^{\max} \geq -p_0$);
- a job i is either sequenced at the left of the top ($x_i^+ = 1$), or at its right ($x_i^- = 1$) or it is not sequenced ($x_i^+ = x_i^- = 1$), hence the constraint $x_i^- + x_i^+ \geq 1$.

The number of late jobs in an admissible solution is $\sum_{i=1}^{n}(x_i^+ + x_i^-) - n$, hence the equivalence between the $\sum U_i$ criterion and the criterion w of the previous problem formulation.

That problem is easy to solve since one can start from the dominant sequence computed by Algorithm 1 and, if it is not admissible, continue to progressively

set the x_i^- and x_i^+ variables to 1, still trying to maximize the increase of $D^{\min} - R^{\max}$, until the sequence becomes admissible. Such a solving procedure is described in Algorithm 2. In the worst case, n additional iterations have to be done until the sequence becomes admissible. So, the time complexity remains $O(n^2)$.

Algorithm 2 An $O(n^2)$ algorithm for finding an optimal sequence with the top 0 on time

Function OptWithTop $(r_i, p_i, d_i,\ i = 0 \dots n)$
$\alpha \prec 0 \prec \beta \leftarrow$ **FindAdm**$(r_i, p_i, d_i,\ i = 0 \dots n)$**;**
$R[i] \leftarrow r_i + \sum_{j \in \alpha} p_j,\ \forall i = 0 \dots n$**;**
$D[i] \leftarrow d_i - \sum_{j \in \beta} p_j,\ \forall i = 0 \dots n$**;**
$R^{\max} \leftarrow \max_{i=0\dots n} R[i]$**;**
$D^{\min} \leftarrow \min_{i=0\dots n} D[i]$**;**
while $(D^{\min} - R^{\max}) < -p_0$ **do**
  $l \leftarrow \mathbf{argmin}_{j \in \alpha} \max(R^{\max} - p_j, \max_{k \le j-1} R[k])$**;**
  $\delta r \leftarrow R^{\max} - \max(R^{\max} - p_l, \max_{k \le l-1} R[k])$**;**
  $r \leftarrow \mathbf{argmax}_{j \in \beta} \min(D^{\min} + p_j, \min_{k \le j-1} D[k])$**;**
  $\delta d \leftarrow \min(D^{\min} + p_r, \min_{k \le r-1} D[k]) - D^{\min}$**;**
  if $(\delta r > \delta d)$ **OR**
  $(\delta r = \delta d$ **AND** $d_0 - R[l] \le D[r] - r_0)$ **then**
    $\alpha \leftarrow \alpha - \{l\}$**;**
    $R[j] \leftarrow R[j] - p_l,\ \forall j = l \dots n$**;**
    $R^{\max} \leftarrow \max_{j=0\dots n} R[i]$**;**
  else
    $\beta \leftarrow \beta - \{r\}$**;**
    $D[j] \leftarrow D[j] + p_r,\ \forall j = r \dots n$**;**
    $D^{\min} \leftarrow \min_{j=0\dots n} R[j]$**;**
  end if
end while
Return $\alpha \prec 0 \prec \beta$**;**

Now, it is easy to understand how to solve the general $1|r_i, nested|\sum U_i$ problem. Indeed, since any optimal sequence has to be compatible with the master sequence $\sigma_\Delta = \{n, n-1, \dots, 1, 0, 1, \dots, n-1, n\}$, one can solve the problem (using Algorithm 2) assuming first that 0 is on time. Then, if the optimal sequence found put at least 1 job late, one can solve the problem assuming that 0 is late and 1 is on time. Then, if the optimal sequence put at least 2 job late, one can solve the problem assuming that 0 and 1 are late and 2 is on time. And so on, until the number of late jobs inside the optimal sequence becomes lower than the number of late tops. Such a procedure is described in Algorithm 3. It works in $O(n^3)$.

Algorithm 3 An $O(n^3)$ optimal algorithm for $1|r_i, nested|\sum U_i$ problems

Function FindOptUi $(r_i, p_i, d_i,\ i = 0 \dots n)$
$U^{\min} \leftarrow \infty$;
$j \leftarrow 0$;
while $j < U^{\min}$ **do**
  $\sigma \leftarrow$ **OptWithTop** $(r_i, p_i, d_i,\ i = j \dots n)$;
  $U \leftarrow$ **Card**(σ);
  if $U + j < U^{\min}$ **then**
    $U^{\min} \leftarrow U + j$;
    $\sigma^* \leftarrow \sigma$;
  end if
  $j \leftarrow j + 1$;
end while
Return σ^*,$U^{\min}$;

7 CONCLUSION

This paper proposes a new $O(n^3)$ procedure for solving $1|r_i, nested|\sum U_i$ problems. That procedure is based on a dominance condition that allows to decide the optimal position of any job relatively to a pivot job (i.e. the top t), provided that the top is on time. To the best of our knowledge, it is the first time that a polynomial procedure is proposed for solving $1|r_i, nested|\sum U_i$ problems. We intend to study now how it can be used for computing new lower bounds for the general $1|r_i|\sum U_i$ problem. We also plan to study the same nested problem, taking interest in minimizing the weighted number of late jobs $(1|r_i, nested|\sum w_i U_i)$.

References

[1] Baptiste P., "Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal", *Journal of Scheduling*, Vol 2., pp245-252, 1999.

[2] Baptiste P., Peridy L and Pinson E., "A Branch and Bound to Mininimze the Number of Late Jobs on a Single Machine with Release Time Constraints", *European Journal of Operational Research*, 144 (1), pp1-11, 2003.

[3] Carlier J., "Problèmes d'ordonnancements à contraintes de Ressources : Algorithmes et Complexité", (*in french*) Thèse d'Etat, Paris VI, 1984.

[4] Dauzère-Pérès S., " Minimizing late jobs in the general one machine scheduling problem", *European Journal or Operations Research*, Vol. 81, pp134-142, 1995.

[5] Dauzère-Pérès S., Sevaux M., "An exact method to minimize the number of

tardy jobs in single machine scheduling", *Journal of Scheduling*, Vol. 7, No 6, pp405-420, 2004.

[6] Erschler J., Roubellat, F., Vernhes J.-P, "Characterizing the set of feasible sequences for n jobs to be carried out on a single machine", *European Journal of Operational Research*, Vol. 4, pp189-194, 1980.

[7] Erschler, J., Fontan, G., Merce, C., Roubellat, F.,"A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates", *Operations Research*, Vol. 31, pp114-127, 1983.

[8] Michael R. Garey, David S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, 1979.

[9] Kise H., Toshihide I., Hisashi M., "A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times", *Operations Research*, 26(1), pp121-12, 1978.

[10] Lawler E.L., "Scheduling a single machine to minimize the number of late jobs", Preprint. Computer Science Division, University of California, Berkeley, 1982.

[11] E.L. Lawler, "A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs", *Ann. Oper. Res.*, 26, pp125-133, 1990.

[12] Michael J. Moore, "An n job, one machine sequencing algorithm for minimizing the number of late jobs", *Management Science*, 15(1), pp102-109, 1968.

[13] Ourari, S., Briand, C., Bouzouia, B. "Une Condition de Dominance pour l'Ordonnancement à une Machine avec Minimisation du Nombre de Travaux en Retard" (*in french*), in *Colloque sur l'Optimisation et les Systèmes d'Information*, Alger, Algeria, 2006.