Minimizing the number of tardy jobs for the single machine scheduling problem: MIP-based lower and upper bounds

Cyril Briand[1,a,b], Samia Ourari[c]

[a] *CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France*
[b] *Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France*
[c] *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. Preemption is not allowed. The objective is to find a feasible job sequence that minimizes the number of tardy jobs. On the basis of an original mathematical integer programming formulation, this paper shows how good-quality lower and upper bounds can be computed. Numerical experiments are provided for assessing the proposed approach.

Keywords: Single machine scheduling, tardy jobs, dominance conditions, mixed-integer programming.

1. INTRODUCTION

A single machine scheduling problem (SMSP) consists of a set V of n jobs to be sequenced on a single disjunctive resource. The interval $[r_j, d_j]$ defines the execution window of each job j, where r_j is the release date of j and d_j, its due-date. The processing time p_j of j is known and preemption is not allowed. A job sequence σ is said feasible if, for any job $j \in V$, $s_j \geq r_j$ and $s_j + p_j \leq d_j$, s_j being the earliest starting time of Job j in σ.

*Corresponding author.
Email addresses: briand@laas.fr (Cyril Briand), sourari@cdta.dz (Samia Ourari)

In this paper, we take an interest in finding a job sequence that minimizes the number of late jobs, problem referred to as $1|r_j|\sum U_j$ in the literature, where U_j is set to 1 if job j is late, *i.e.*, $U_j \leftarrow (s_j + p_j > d_j)$. In the sequel, we review some important papers that deal with this problem. Nevertheless, the literature is also rich of papers that consider other important variants by considering various additional assumptions, such as : jobs are weighted (problem $1|r_j|\sum w_j U_j$), setup times are considered, there exists several identical machines, *etc.* For a more extended review, the reader is referred to [5].

Determining whether it exists a feasible sequence, *i.e.*, all jobs meet their due dates, is NP-complete [14]. The problem of minimizing the number of late jobs is also NP-hard [10]. Efficient branch-and-bound procedures are reported in [2, 8, 5] that solve problem instances with up to 200 jobs.

When additional assumptions are made, $1|r_j|\sum U_j$ problems can become solvable in polynomial time. For instance, when release dates are equal, $1||\sum U_j$ problems can be solved in $\mathrm{O}(n\log(n))$ using Moore's well known algorithm [15]. Considering the case where release and due dates of jobs are similarly ordered, *i.e.*, $r_i < r_j \Rightarrow d_i \leq d_j$, Kise, Ibaraki and Mine proposed a dynamic programming algorithm that runs in $\mathrm{O}(n^2)$ [11]. Under this same assumption, an $\mathrm{O}(n\log(n))$ algorithm was later proposed by Lawler in 1982 [12]. Lawler [13] also described an $\mathrm{O}(n\log(n))$ algorithm that works on preemptive nested problems $1|r_j, nested, pmtn|\sum U_j$, *i.e.*, job preemption is allowed and job execution windows are nested: $r_i < r_j \Rightarrow d_i \geq d_j$ or $d_i > d_j \Rightarrow r_i \leq r_j$. More recently, considering the general preemptive problem $1|r_j, pmtn|\sum U_j$, Baptiste designed an algorithm that runs in $\mathrm{O}(n^4)$ [1]. When processing times are equal, Carlier [6] proposed in the early eighties a $\mathrm{O}(n^3\log(n))$ procedure. Nevertheless, this procedure has recently been proved incorrect by Chrobak and al. [7] who exhibit a new optimal $\mathrm{O}(n^5)$ algorithm.

This paper presents how, using an original Mixed Integer Programming (MIP) formulation, both good-quality lower and upper bounds can be computed for the $1|r_j|\sum U_j$ problem. The formulation is inspired by the one described in [4] for solving the $1|r_j|L_{\max}$. Another advantage of our approach precisely lies

in its nature since, thanks to the genericity of MIP, additional user-preference constraints can easily be inserted within the model. Let also remark that, even if only the minimization of tardy jobs is discussed below, our approach remains valid for the more general $1|r_j|\sum w_jU_j$ since only the objective function has to be adapted (which is quite straightforward), all the constraints remaining unchanged.

The paper is structured as follows. First, considering the problem of finding a feasible sequence to the SMSP, a dominance theorem is recalled. In Section 3, the MIP formulation proposed in [4] for searching a feasible job sequence is detailed. The fourth section discusses the validity of the dominance theorem stated in Section 2 for the case of the $\sum U_j$ criterion. Section 5 shows how the MIP of Section 3 can be adapted for computing upper bounds and lower bounds to the $1|r_j|\sum U_j$ problem. The last section is devoted to the analysis of our computational experiments.

2. A dominance theorem for the SMSP

In this section, some analytical dominance conditions are recalled for the SMSP. They have been originally proposed in the early eighties by Erschler et al. [9] within a theorem that is stated in the sequel. This theorem uses the notions of a *top* and a *pyramid* that are defined below. It defines a set S_{dom} of dominant job sequences, with respect to the feasibility problem, for the SMSP. Let us recall that a job sequence σ_1 dominates another job sequence σ_2 if σ_2 feasible $\Rightarrow$ σ_1 feasible. By extension, a set of job sequences S_{dom} is said dominant if, for any job sequence $\sigma_2 \notin S_{\text{dom}}$, it exists $\sigma_1 \in S_{\text{dom}}$ such that σ_2 feasible $\Rightarrow$ σ_1 feasible.

Characterizing a set of dominant job sequences is of interest since, when searching for a feasible job sequence, only the set of dominant sequences need to be explored. Indeed, when there does not exist any feasible sequence in the dominant set, it can be asserted that the original problem does not admit any feasible solution. This allows a significant reduction of the search space.

Definition 1. *A job $t \in V$ is a top if there does not exist any other job $j \in V$ such that $r_j > r_t \wedge d_j < d_t$ (i.e., the execution window of a top does not include (strictly) any other execution window).*

The tops are indexed in ascending order with respect to their release dates or, in case of tie, in ascending order with respect to their due dates. When both their release dates and due dates are equal, they can be indexed in an arbitrary order. Thus, if t_a and t_b are two tops then $a < b$ if and only if $(r_{t_a} \leq r_{t_b}) \wedge (d_{t_a} \leq d_{t_b})$. Let refers to $\mathcal{T}$ as the set of tops, with cardinality $|\mathcal{T}| = m$.

Definition 2. *Given a top t_k, a pyramid P_k related to t_k is the set of jobs $j \in V \setminus \mathcal{T}$ such that $r_j < r_{t_k} \wedge d_j > d_{t_k}$ (i.e., the set of jobs whose execution window strictly includes the execution window of the top).*

Considering the previous definition, let remark that a non-top job can belong to several pyramids. Let $u(j)$ ($v(j)$ resp.) refers to the index of the first pyramid (the last pyramid resp.) to which Job j can be assigned.

The following theorem can now be stated. The reader is referred to [9] for its proof.

Theorem 1. *The set S_{dom} of job sequences in the form:*

$$\alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_k \prec t_k \prec \beta_k \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$$

is dominant for the problem of finding a feasible job sequence. Where:

- *t_k is the top of Pyramid P_k, $\forall k = 1 \ldots m$;*
- *α_k and β_k are two job subsequences belonging to P_k and located at the left and the right of Top t_k respectively, such that jobs belonging to subsequence α_k are sequenced with respect to the increasing order of their r_j, and jobs belonging to β_k, are sequenced with respect to the increasing order of their d_j ;*
- *any non-top j is located either in subsequence α_k or β_k, for a given k such that $u(j) \leq k \leq v(j)$.*

In the previous theorem, it must be pointed out that the numerical values of the processing times p_j as well as those 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. In order to illustrate the theorem, let us consider an instance of seven jobs, so that the relative order among the release dates r_j and the due dates d_j of the jobs is $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 < (r_5 = r_7) < d_6 < d_5 < d_1 < d_7$. This instance has four top jobs : $t_1 = 2$, $t_2 = 4$, $t_3 = 5$ and $t_4 = 7$, which characterize four pyramids: $P_1 = \{1, 3, 6\}$, $P_2 = \{1, 6\}$, $P_3 = \{1\}$ and $P_4 = \emptyset$ (in accordance with the definition, a top job does not belong to the pyramid it characterizes). According to Theorem 1, whatever the real values of r_i and d_i (provided they remain compatible with the previous interval structure), the dominant sequences are in the form $\alpha_1 \prec 2 \prec \beta_1 \prec \alpha_2 \prec 4 \prec \beta_2 \prec \alpha_3 \prec 5 \prec \beta_3 \prec 7$, where jobs belonging to subsequence α_k (β_k respectively) are sequenced with respect to the increasing order of their r_j (d_j respectively). Thus, for this example we have $\alpha_1 \in \{6 \prec 1 \prec 3\}$, $\beta_1 \in \{3 \prec 6 \prec 1\}$, $\alpha_2 = \emptyset$, $\beta_2 \in \{6 \prec 1\}$, $\alpha_3 = \emptyset$, $\beta_3 \in \{1\}$, $\alpha_4 = \emptyset$ and $\beta_4 = \emptyset$, and 24 dominant sequences (out of $7! = 5040$) are characterized :

$6 \prec 1 \prec 3 \prec 2 \prec 4 \prec 5 \prec 7$	,	$1 \prec 3 \prec 2 \prec 6 \prec 4 \prec 5 \prec 7$
$1 \prec 3 \prec 2 \prec 4 \prec 6 \prec 5 \prec 7$	,	$6 \prec 1 \prec 2 \prec 3 \prec 4 \prec 5 \prec 7$
$1 \prec 2 \prec 3 \prec 6 \prec 4 \prec 5 \prec 7$	,	$1 \prec 2 \prec 3 \prec 4 \prec 6 \prec 5 \prec 7$
$6 \prec 3 \prec 2 \prec 1 \prec 4 \prec 5 \prec 7$	,	$3 \prec 2 \prec 6 \prec 1 \prec 4 \prec 5 \prec 7$
$3 \prec 2 \prec 1 \prec 4 \prec 6 \prec 5 \prec 7$	,	$6 \prec 2 \prec 3 \prec 1 \prec 4 \prec 5 \prec 7$
$2 \prec 3 \prec 6 \prec 1 \prec 4 \prec 5 \prec 7$	,	$2 \prec 3 \prec 1 \prec 4 \prec 6 \prec 5 \prec 7$
$6 \prec 3 \prec 2 \prec 4 \prec 1 \prec 5 \prec 7$	,	$3 \prec 2 \prec 6 \prec 4 \prec 1 \prec 5 \prec 7$
$3 \prec 2 \prec 4 \prec 6 \prec 1 \prec 5 \prec 7$	,	$6 \prec 2 \prec 3 \prec 4 \prec 1 \prec 5 \prec 7$
$2 \prec 3 \prec 6 \prec 4 \prec 1 \prec 5 \prec 7$	,	$2 \prec 3 \prec 4 \prec 6 \prec 1 \prec 5 \prec 7$
$6 \prec 3 \prec 2 \prec 4 \prec 5 \prec 1 \prec 7$	,	$3 \prec 2 \prec 6 \prec 4 \prec 5 \prec 1 \prec 7$
$3 \prec 2 \prec 4 \prec 6 \prec 5 \prec 1 \prec 7$	,	$6 \prec 2 \prec 3 \prec 4 \prec 5 \prec 1 \prec 7$
$2 \prec 3 \prec 6 \prec 4 \prec 5 \prec 1 \prec 7$	,	$2 \prec 3 \prec 4 \prec 6 \prec 5 \prec 1 \prec 7$

3. A MIP formulation for finding a feasible job sequence

In this section, the problem of searching a feasible job sequence is considered and a MIP is described, which has been originally introduced in [4]. It aims at determining the most dominant job sequence among the set S_{dom} that is in the form $\alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$. Let us highlight that the case where

a non-top job is sequenced at the right of a top k, is strictly equivalent to the case where this job has been sequence at the left of top $k+1$. These cases are not distinguished in the MIP model below.

The MIP is formulated as follows:

$$\max \; z = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k})$$

$$\text{s.t.} \quad \begin{cases} R_k \geq r_{t_k} & , \; \forall t_k \in \mathcal{T} & (3.1) \\ R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_{k-1,j} & , \; \forall i \in V \setminus \mathcal{T}, u(i) = k & (3.2) \\ R_k \geq R_{k-1} + p_{t_{k-1}} + \sum_{j \in P_{k-1}} p_j x_{k-1,j} & & \\ \quad + \sum_{u(j)=k} p_j x_{k-1,j} & , \; \forall t_k \in \mathcal{T} \setminus \{t_1\} & (3.3) \\ D_k \leq d_{t_k} & , \; \forall t_k \in \mathcal{T} & (3.4) \\ D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_{k,j} & , \; \forall i \in V \setminus \mathcal{T}, v(i) = k & (3.5) \\ D_k \leq D_{k+1} - p_{t_{k+1}} - \sum_{u(j)=k+1} p_j x_{kj} & & \\ \quad - \sum_{j \in P_k} p_j x_{kj} & , \; \forall t_k \in \mathcal{T} \setminus \{t_m\} & (3.6) \\ \sum_{k=u(i)-1}^{v(i)} x_{ki} = 1 & , \; \forall i \in V \setminus \mathcal{T} & (3.7) \\ x_{ki} \in \{0,1\} & , \forall i \in V \setminus \mathcal{T}, \forall k \in [u(i)-1; \; v(i)] & \\ D_k, R_k \in \mathbb{Z} & , \; \forall t_k \in \mathcal{T} & \end{cases}$$

In this MIP, a binary variable x_{ki}, with $k = u(i) - 1$ to $v(i)$, is assigned to each non-top job i. If i is sequenced at the left of top $u(i)$ then $x_{u(i)-1,i} = 1$, otherwise $x_{u(i)-1,i} = 0$. Now, if job i is sequenced at the right of any top $k \in [u(i), v(i)]$ then $x_{ki} = 1$, otherwise $x_{ki} = 0$. Constraints 3.7 ensure that i is sequenced once.

The integer variable R_k gives the earliest starting time of Job t_k. By definition, we know that:

$$R_k = \max(r_{t_k}, eft_{k-1} + \sum_{\{j \in \alpha_k\}} p_j, \max_{i \in \alpha_k}(r_i + p_i + \sum_{\{j \in \alpha_k | i \prec j\}} p_j)) \qquad (3.8)$$

where eft_{k-1} is the earliest completion time of the job subsequence β_{k-1}. As the variable R_{k-1} corresponds to the earliest starting time of Job t_{k-1}, it comes that $eft_{k-1} = R_{k-1} + p_{t_{k-1}} + \sum_{j \in \beta_{k-1}} p_j$. Therefore, the constraints (3.1), (3.2) and (3.3), according to Equation 3.8, allow to determine the value of R_k.

Symmetrically, the integer variable D_k corresponds to the latest finishing time of Job t_k. By definition:

$$D_k = \min(d_{t_k}, lst_{k+1} - \sum_{\{j \in \beta_k\}} p_j, \min_{i \in \beta_k}(d_i - p_i - \sum_{\{j \in \beta_k | j \prec i\}} p_j)) \qquad (3.9)$$

where lst_{k+1} is the latest starting time of the job subsequence α_{k+1}. As the variable D_{k+1} corresponds to the latest finishing time of Job t_{k+1}, it comes that $lst_{k+1} = D_{k+1} - p_{t_{k+1}} - \sum_{j \in \alpha_{k+1}} p_j$. Therefore, the constraints (3.4), (3.5) and (3.6), according to Equation 3.9, give to D_k its value.

Obviously, it can be observed that the values of the R_k and D_k variables of the MIP can directly be deduced from the values of the x_{ki} binary variables. In [4], it is shown that if $z = \min_{k=1,\ldots,m}(D_k - R_k - p_{t_k})$ is maximized while respecting the constraints, then the obtained sequence dominates all the others. Indeed, the authors give the proof that, for any feasible sequence, the maximum lateness $L_{\max}$ strictly equals $-z$. Therefore, maximizing z is strictly equivalent to minimizing the maximum lateness $L_{\max}$ and it can be asserted that any sequence $\alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$ is feasible if and only if $z = \min_{k=1,\ldots,m}(D_k - R_k - p_{t_k}) \geq 0$. In the case where $z^* < 0$, there obviously does not exist any feasible sequence of n jobs for the considered problem.

For illustration, let consider an instance of 5 jobs with the relative order among releases dates and dues dates : $r_3 < r_1 < r_{t_1} < d_{t_1} < d_1 < d_3 < r_2 < r_{t_2} < d_{t_2} < d_2 < d_3$, i.e., $P_{t_1} = \{1, 3\}$ and $P_{t_2} = \{2, 3\}$. If the solution of the MIP is $x_{01} = 1, x_{11} = 0; x_{03} = 0, x_{13} = 1, x_{23} = 0$, and $x_{12} = 0$, $x_{22} = 1$ then the most dominant sequence is: $1 \prec t_1 \prec 3 \prec t_2 \prec 2$.

4. Dominance condition for $1|r_j|\sum U_j$

In this section, the $\sum U_j$ criterion is considered. Searching an optimal solution for $1|r_j|\sum U_j$ problem amounts to determine a feasible sequence for the largest selection of jobs $E \subseteq V$. Let E^* be this 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 a feasible job sequence for on-time jobs. Consequently, Theorem 1 can be applied only to the jobs belonging to E^*. There are $\sum_{k=1\ldots n} C_n^k$ possible different selections of jobs. Regarding the $\sum U_j$ criterion, the following corollary is proved.

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

Note that, clearly, Theorem 1 remains also valid regarding the $\sum w_j U_j$ criterion.

As already pointed out, the number of possible job selections is quite large. Nevertheless, as explained in [16], it is not necessary to enumerate all the possible job selections to get the dominant sequences. Indeed, they can be characterized using one or more *master-pyramid sequences*. The notion of a master-pyramid sequence is somewhat close to the notion of a master sequence that Dauzères-Pérès and Sevaux proposed in [8]. It allows to easily verify whether a job sequence belongs to a set of dominant sequences. For building up a master-pyramid sequence associated with a job selection $E \subseteq V$, the number m_E of tops and pyramids have first to be determined. Then, the set of dominant sequences being in the form $\alpha_1(E) \prec t_1(E) \prec \beta_1(E) \prec \cdots \prec \alpha_k(E) \prec t_k(E) \prec \beta_k(E) \prec \cdots \prec \alpha_{m_E}(E) \prec t_{m_E}(E) \prec \beta_{m_E}(E)$, we assign any non-top job j in sequence $\alpha_k(E)$, if $k = u(j)$, and in all the sequences $\beta_k(E)$ such that $u(j) \leq k \leq v(j)$. The $\alpha_k(E)$ and $\beta_k(E)$ subsequences are ordered with respect to Theorem 1.

For illustration, let us consider the problem instance with 7 jobs given in the previous section. The master-pyramid sequence associated with the selection $E = V$ is (tops are in bold):

$$\sigma_\Delta(V) = (6 \prec 1 \prec 3) \prec \mathbf{2} \prec (3 \prec 6 \prec 1) \prec \mathbf{4} \prec (6 \prec 1) \prec \mathbf{5} \prec (1) \prec \mathbf{7}$$

Any job sequence of n jobs *compatible* with $\sigma_\Delta(V)$ belongs to the set of dominant sequences. A sequence s is said compatible with the master-pyramid sequence $\sigma_\Delta(V)$ if the order of the jobs in s does not contradict the possible orders defined by $\sigma_\Delta(V)$, this will be denoted as $s \subset \sigma_\Delta(V)$.

Theorem 2. *Considering a given problem $1|r_j|\sum U_j$ with the set of jobs V, if an optimal solution with all tops on-time exists then, $\sigma_\Delta(V)$ characterizes a set of dominant sequences that contains the optimal solution.*

Proof. Under the hypothesis that all tops are on-time, it is obvious that $\sigma_\Delta(V)$ also characterizes the set of dominant sequences of any job selection E such that $\{t_1, \ldots, t_m\} \subseteq E$. In other words, if s is a job sequence such that $s \subset \sigma_\Delta(E)$ then $s \subset \sigma_\Delta(V)$. □

Nevertheless, $\sigma_\Delta(V)$ does not necessarily characterize all the job sequences being dominant for the $\sum U_j$ criterion. This assertion can easily be illustrated considering a problem V with 4 jobs having the interval structure, represented on Figure 1. Job t is the top of the structure and the master-pyramid sequence $\sigma_\Delta(V)$ is $(c \prec b \prec a) \prec \mathbf{t} \prec (a \prec b \prec c)$. Now, let us suppose that it does not exist an optimal solution with t on-time (the interval of t can be ignored). In this case, $E = V \setminus \{t\} = \{a, b, c\}$ and there are two tops a and b with a master-pyramid sequence $\sigma_\Delta(E) = (c \prec \mathbf{a} \prec c \prec \mathbf{b} \prec c)$. As it can be observed, $\sigma_\Delta(E)$ is not compatible with $\sigma_\Delta(V)$ since, in the former, Job c cannot be sequenced between a and b, while this is possible in the latter, i.e., $(\mathbf{a} \prec c \prec \mathbf{b}) \subset \sigma_\Delta(E)$ is not compatible with $\sigma_\Delta(V)$. This simple example shows that the complete characterization of the set of dominant sequences requires to enumerate all the non-compatible master-pyramid sequences, their number being possibly exponential in the worst case.

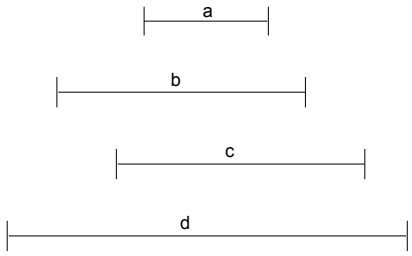

Figure 1: An interval structure

Up to now, we have shown that Theorem 1 is in general not dominant regarding $1|r_j|\sum U_j$ criterion. In the sequel, we focus on a particular SMSP structure where any pyramid P_k, $\forall k = 1, \ldots, m$ is *perfect*. A pyramid P_k is said perfect if, $\forall (i,j) \in P_k \times P_k$, $(r_i \geq r_j) \Leftrightarrow (d_i \leq d_j)$, *i.e.*, the execution intervals of the jobs belonging to P_k are included each inside the other. By extension, when all the pyramids are perfect, the corresponding SMSP will be said perfect. For this special case, the following theorem is proved:

Theorem 3. *Considering a perfect SMSP with job set V, the master-pyramid sequence $\sigma_\Delta(V)$ characterizes a set of dominant sequences for the $\sum U_j$ criterion.*

Proof. Obviously, removing a job j from a perfect SMSP V produces a new perfect SMSP $V \setminus \{j\}$. The proof goes by showing that the master pyramid sequence $\sigma_\Delta(V \setminus \{j\})$ is compatible with $\sigma_\Delta(V)$ (in other words, all the sequences that are compatible with $\sigma_\Delta(V \setminus \{j\})$ are also compatible with $\sigma_\Delta(V)$). Let us assume first that the removed job j is a non-top job. Since $\sigma_\Delta(V)$ is built up according to the position of the tops, removing j from V induces to remove j from all the subsequences α_k, β_k of $\sigma_\Delta(V)$ such that $u(j) \leq k \leq v(j)$ and, in this case, the relation $\sigma_\Delta(V \setminus \{j\}) \subset \sigma_\Delta(V)$ necessarily holds.
Let us assume now that j is a top having the index x. The master-pyramid sequence σ_Δ is in the form $(\alpha_1, t_1, \ldots, t_{x-1}, \beta_{x-1}, \alpha_x, j, \beta_x, \alpha_{x+1}, t_{x+1}, \ldots, t_m, \beta_m)$.

Two cases have to be considered: if j is a top such that $\forall i \in P_x \Rightarrow i \in P_{x-1}$ or $i \in P_{x+1}$, then $\sigma_\Delta(V \setminus \{j\}) = (\alpha_1, t_1, \ldots, t_{x-1}, \beta_{x-1}, \alpha_{x+1}, t_{x+1}, \ldots, t_m, \beta_m)$ and in this case, $\sigma_\Delta(V \setminus \{j\})$ is obviously compatible with $\sigma_\Delta(V)$. Otherwise, let k be the last job of α_x (*i.e.*, $\alpha_x = (\alpha'_x, k)$). Since the execution intervals of the jobs belonging to P_x are included each inside the other, the order of the jobs in α_x matches the reverse order of the jobs of β_x, therefore k is also the first job of β_x (*i.e.*, $\beta_x = (k, \beta'_x)$). Then $\sigma_\Delta(V \setminus \{j\}) = (\alpha_1, t_1 \ldots t_{x-1}, \beta_{x-1}, \alpha'_x, k, \beta'_x, \alpha_{x+1}, t_{x+1} \ldots t_m, \beta_m)$ and in this case, $\sigma_\Delta(V \setminus \{j\})$ is obviously compatible with $\sigma_\Delta(V)$. □

5. A MIP formulation for the $1|r_j|\sum U_j$ problem

The MIP of Section 2, which returns the most dominant sequence, can easily be adapted for solving either the problem $1|r_j|\sum U_j$ under the hypothesis that tops are on time or the perfect SMSP case. The new MIP is described below and differs from the previous one only by the addition of the terms in bold. Allowing a non-top job to be late is easy by relaxing constraints (3.7), replacing it by constraints (5.7). As the feasibility of the obtained sequence is required, the constraints $D_k - R_k \geq p_{t_k}$ is set, $\forall k = 1, \ldots, m$. Nevertheless, these constraints can be too strong since, when two consecutive tops t_k and t_{k+1} are such that $d_{t_{k+1}} - r_{t_k} < p_{t_{k+1}} + p_{t_k}$, the MIP is unfeasible (*i.e.*, there does not exist any feasible sequence that keeps both t_k and t_{k+1} on time). For avoiding this unfeasibility, the binary variable y_k is introduced (see constraints (5.8)) such that y_k equals 1 if the processing time of t_k is ignored (t_k is late), 0 otherwise.

As a top can be late, constraints (3.3) and (3.6) must be adapted (see constraints (5.3) and (5.6)) since the duration p_{t_k} should not be taken into account when $y_k = 1$. Moreover, still in the case where t_k is late, constraints (3.1) and (3.4) should be relaxed: this can be done by replacing them by constraints (5.1) and (5.5), respectively, with $R_0 = \min_{j \in V \setminus \mathcal{T}} r_j$ and $D_{m+1} = \max_{j \in V \setminus \mathcal{T}} d_j$. Indeed, constraint $R_k \geq R_0$ and $D_k \leq D_{m+1}$ are always valid. On the other hand, constraints (3.2)-(3.4) should also be relaxed in the case job i is late.

This can be done in the same way as before, using constraints (5.2) and (5.4.), respectively. Finally, the $\sum U_j$ criterion can easily be expressed by using the binary variables y_k and x_{ki} since if $\sum_{k=u(j)-1}^{v(j)} x_{jk} = 0$, the non-top job j is not assigned to any pyramid and can be considered late.

$$\min \; z = \sum_{i \in V \setminus \{t_1 \ldots t_m\}} (1 - \sum_{k=u(i)-1}^{v(i)} x_{ki}) + \sum_{k=1}^{m} y_k$$

s.t.

$$\begin{cases}
R_k \geq r_{t_k} + \mathbf{y_k}(\mathbf{R_0} - \mathbf{r_{t_k}}) & , \; \forall t_k \in \mathcal{T} & (5.1) \\
R_k \geq r_i + (\mathbf{1} - \mathbf{x_{k-1,i}})(\mathbf{R_0} - \mathbf{r_i}) & & \\
\quad + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_{k-1,j} & , \; \forall i \in V \setminus \mathcal{T}, u(i) = k & (5.2) \\
R_k \geq R_{k-1} + p_{t_{k-1}}(\mathbf{1} - \mathbf{y_{k-1}}) & & \\
\quad + \sum_{j \in P_{k-1}} p_j x_{k-1,j} & & \\
\quad + \sum_{u(j)=k} p_j x_{k-1,j} & , \; \forall t_k \in \mathcal{T} \setminus \{t_1\} & (5.3) \\
D_k \leq d_{t_k} + \mathbf{y_k}(\mathbf{D_{m+1}} - \mathbf{d_{t_k}}) & , \; \forall t_k \in \mathcal{T} & (5.4) \\
D_k \leq d_i + (\mathbf{1} - \mathbf{x_{ki}})(\mathbf{D_{m+1}} - \mathbf{d_i}) & & \\
\quad - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_{k,j} & , \; \forall i \in V \setminus \mathcal{T}, v(i) = k & (5.5) \\
D_k \leq D_{k+1} - p_{t_{k+1}}(\mathbf{1} - \mathbf{y_{k+1}}) & & \\
\quad - \sum_{u(j)=k+1} p_j x_{kj} & & \\
\quad - \sum_{j \in P_k} p_j x_{kj} & , \; \forall t_k \in \mathcal{T} \setminus \{t_m\} & (5.6) \\
\sum_{k=u(i)-1}^{v(i)} x_{ki} \leq 1 & , \; \forall i \in V \setminus \mathcal{T} & (5.7) \\
D_k - R_k \geq p_{t_k}(\mathbf{1} - \mathbf{y_k}) & , \; \forall t_k \in \mathcal{T} & (5.8) \\
y_k \in \{0,1\} & \forall t_k \in \mathcal{T} & \\
x_{ki} \in \{0,1\} & , \forall i \in V \setminus \mathcal{T}, \forall k \in [u(i)-1; \; v(i)] & \\
D_k, R_k \in \mathbb{Z} & , \; \forall t_k \in \mathcal{T} &
\end{cases}$$

Obviously, considering any problem V, solving the previous MIP gives an upper bound for yhe number of tardy jobs, *i.e.*, it returns the best solution found in $\sigma_\Delta(V)$. Nevertheless, since the set of sequences compatible with $\sigma_\Delta(V)$ is not dominant, there is not guaranty for this solution to be optimal.

On the other hand, a lower bound can be obtained by relaxing some constraints and optimally solving the relaxed problem. From Theorem 3, when the problem is perfect, we know that the set of sequences in the form $\alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$ is dominant for the $\sum U_j$ criterion. Furthermore, considering any problem, it is always possible to decrease the r_j values (or increase the d_j values) of some jobs in order to make it perfect, *i.e.*, $\forall (i,j) \in P_k \times P_k$, $(r_i \geq r_j) \Leftrightarrow (d_i \leq d_j)$, $\forall k = 1, \ldots, m$ (see Figure 2). Doing so, a relaxed problem is obtained that can be optimally solved by the last MIP. This will give us a lower bound for the number of tardy jobs.

Of course, in the case where both lower and upper bounds are equal, it can directly be deduced that the solution found for the upper bound is optimal.

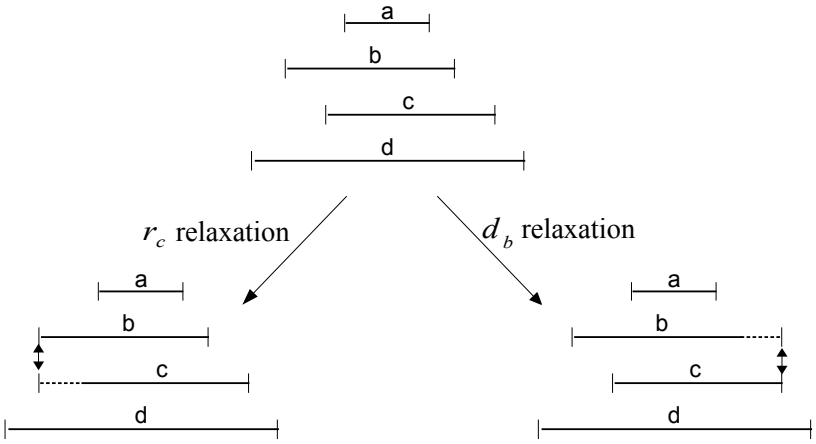

Figure 2: Two relaxation strategies for converting a problem into a perfect one

6. Numerical experiments

For evaluating the performances of our MIP model, Baptiste et al.'s problem instances have been used (see [2]). For $n \in \{80, 100, 120, 140, 160\}$, 120 problem instances are provided and, for each of them, thanks to the authors who provide us with their detailed results, either the optimal $\sum U_j$ value (OPT) or, at least, an upper-bound of this value ($BEST$), is known.

For each problem instance, using a commercial MIP solver, we determined one lower bound (LB) and one upper bound (UB). For the lower bound, we chose to relax the deadlines of the non-top jobs to convert the problem into a perfect one, then to solve the corresponding MIP. Indeed, from preliminary experiments, it turns out that relaxing the d_i give in general better result than relaxing the release times r_i. Since the two kinds of relaxation are symmetric, this is possibly due to the way problem instances have been generated (see [2]). The upper bound is obtained by directly solving the MIP using the initial r_i and d_i values.

In each run, the CPU time has been bounded to one hour. Table 1 displays, for each class of problem, the percentages of LB and UB that were proven optimal within one hour, as well as the min / mean / max CPU time. For example, when $n = 80$, the solver returns the optimal LB value in 100% of the cases, with a min / mean / max CPU time of 0.01/8.075/761 seconds respectively. The values of Table 1 can be compared with the percentage of time the Baptiste et al.'s method finds an optimal solution in less than one hour (see Table 3).

A few observations can be made at this point. First, even if some problem instances seem very hard to solve (the proof of optimality is very time-consuming), optimal solutions are found in most of the cases, the computation time being rather low. We also observe that the computation of the upper bound is globally less time expensive than the one of the lower bound. One explanation could be that, in the former case, because tops are assumed on time, the search space is less extended than in the latter case, where any job can be late or on time.

Even if finding optimal solution is not the aim of our approach, Table 2 takes

Problem class		*LB*	*UB*
n=80	*N*	100.00%	99.16%
	Tcpu	(0.01; 8.075; 761) s	(0.01; 1.40; 113) s
n=100	*N*	97.5%	99.16%
	Tcpu	(0.02; 10.44; 460) s	(0.02; 5.96; 553) s
n=120	*N*	97.5%	93.33%
	Tcpu	(0.02; 54.18; 2554) s	(0.02;25.04; 858) s
n=140	*N*	92.5%	86.66%
	Tcpu	(0.05; 28.31;721) s	(0.02 ; 22.78 ; 905) s
n=160	*N*	84.16%	82.5%
	Tcpu	(0.05; 81.03; 3067) s	(0.02; 86.37; 2496) s

Table 1: Percentages of MIP solved to optimality

Problem class	*LB=UB*	*UB - LB*
	All instances	min; mean; max
n=80	59.67%	0; 0.58 ; 3
n=100	52.94%	0; 0.69; 6
n=120	50.9%	0; 0.78; 5
n=140	51.2%	0; 0.93; 5
n=160	35.35%	0; 1.14; 6

Table 2: Percentages of optimal solutions

an interest in the cases where the solution UB can be directly proved optimal, *i.e.*, $(UB = LB)$. As one can see, optimality can be proved in many cases, even if ad-hoc approaches remain better suited from this point of view. Furthermore, the gap between LB and UB is very tight in average. Let us point out that our MIP approach proves the optimality of 48 instances that were not optimally solved by Baptiste et al.. Moreover, Baptiste et al. did not succeed to solve most of the problems with 160 jobs (only 15), while we solved 101 of them and proved the optimality of 35.

Lastly, Table 3 gives a more tightened analysis of the quality of our lower bounds LB and upper bounds UB. LB bounds are systematically compared with the optimal value OPT found by the Baptiste et al.'s method (when it is computed in less than one hour). We observe than LB have a good quality since

they directly equals OPT in 90% of the cases. The bounds UB are even better since they are lower or equal to $BEST$ in 98% of the cases. These observations seems to indicate that, in order to increase the percentage of solutions that our approach is able to certify optimal, the way to relax the problem for finding lower bounds should be improved. Mixed relaxation schemes where r_i and d_i values would be both relaxed, intending to minimize the sum of their variations, could be investigated.

Problem classes	*Tcpu*<3600s and LB=OPT	UB<=BEST
all	90.9%	98.2%

Table 3: Analysis of the lower and upper bounds quality

7. Conclusion

Designing MIP models for solving efficiently basic SMSPs is of interest since MIP approaches can be more easily adapted when dealing with new constraints or new objective. As a proof of this statement, this paper shows how an original MIP model, used for solving the $1|r_j|L_{\max}$ problem, can be adapted for dealing with the more complex $1|r_j|\sum U_j$ problem. Since the analytical dominance condition used for designing the MIP formulation of the former problem is valid for the $\sum U_j$ criterion only under some restrictions (tops are not late), just an upper bound can be computed. However, as shown by the experiments, this upper bound is optimal in most of the cases, or at least very close to the optimum. In the particular case where the considered SMSP is *perfect* (see Section 4), our MIP gives an optimal $\sum U_j$ value. Since it is always possible to relax the release dates or the due dates of any SMSP in order to make it perfect, this MIP also allows to compute good-quality lower bounds.

For the future works, we plan to investigate preprocessing methods by applying variable-fixing techniques from Integer Linear Programming. Such techniques, successfully used in several papers (see for instance [3]), would allow to definitively fix the value of some binary variables, namely those of y_{t_k}, x_{ki} in our

MIP, intending to tighten the search space and speed up the solving phase. We believe that these techniques could improve our approach from a computational viewpoint such it can hopefully become competitive with existing branch and bound procedures.

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] Baptiste P., Della Croce F., Grosso A. and T'kindt V. (2009), "Sequencing a single machine with due dates and deadlines : an ILP-based approach to solve very large instances", Journal Of Scheduling, ISSN 1099-1425 (Online).

[4] Briand C., Ourari S., Bouzouia B (2010), "An efficient ILP formulation for the single machine scheduling problem ", RAIRO-Operations Research, Vol 44, no 1, pp 61-71, 2010.

[5] R. M'Hallah, R.L. Bulfin "Minimizing the weighted number of tardy jobs on a single machine with release dates", European Journal of Operational Research, 176, pp727-744 (2007).

[6] Carlier J., "Problèmes d'ordonnancements à durées égales", QUESTIO, 5(4), 219-228 (1981) (*in french*).

[7] Chrobak M., Dürr C., Jawor W., Kowalik L., Kurowski M., "A Note on Scheduling Equal-Length Jobs to Maximize Throughput", Journal of Scheduling, Vol. 9, No 1, pp71-73 (2006).

[8] 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).

[9] 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.

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

[11] Kise H., Toshihide I., Mine H., "A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times", Operations Research, 26(1), pp121-126 (1978).

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

[13] 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).

[14] Lenstra J.K., Rinnooy Han A.H.G , Brucker P., "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, vol. 1, pp343-362,1977.

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

[16] Ourari S., Briand C. "Conditions de dominance pour le problème à une machine avec minimisation des travaux en retard" 9ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'08), Clermont-Ferrand (France), pp.351-352 (2008)(*in french*).