# Mathematical Integer Programming for a One Machine Scheduling Problem

Samia Ourari[1,2] and Cyril Briand[2], and Brahim Bouzouia[1]

[1]Centre de Développement des Technologies Avances, Alger, Algrie
[2]Laboratoire d'Architecture et d'Analyses des Systmes-CNRS; Toulouse, France
sourari@cdta.dz;briand@laas.fr;bbouzouia@cdta.dz

**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 both good-quality lower and upper bounds can be computed. Numerical experiments on Baptiste et al.'s instances are provided, which demonstrate the efficiency of the approach.

## 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 $\sigma$ 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 $\sigma$.

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 litterature, 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 litterature 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 $O(n \log(n))$ 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 $O(n^2)$ [11]. Under this same assumption, an $O(n \log(n))$ algorithm was later proposed by Lawler in 1982 [12]. Lawler [13] 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 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 $O(n^4)$ [1]. When processing times are equal, Carlier [6] proposed in the early eighties a $O(n^3 \log(n))$ procedure. Nevertheless, this procedure has recently been proved non-optimal by Chrobak et al. [7] who exhibit a new optimal $O(n^5)$ algorithm.

This paper presents how, using an original Mathematical Integer Programming (MIP) formulation, both good-quality lower and upper bounds can be computed for the $1|r_j|\sum U_j$ problem. The proposed approach mainly differs from the branch-and-bound approaches described in [2], [8] and [5] in the fact that, since MIP is used, it is more generic: new constraints can easily be added to the model. Moreover, from the best of our knowledge, there does not exist other MIP approach for the $1|r_j|\sum U_j$ problem that can be compared in terms of efficiency with the already existing branch-and-bound methods.

The paper is structured as follows. First, a dominance theorem is recalled that stands for the problem of finding a feasible sequence to the SMSP. In Section 3, a MIP formulation for searching a feasible job sequence is described. The fourth section discusses the validity of the dominance theorem of Section 2 when the $\sum U_j$ criterion is considered. Section 5 and 6 show how the MIP of Section 3 can easily be adapted for computing an upper bound and a lower bound to the $1|r_j|\sum U_j$ problem. The last section is devoted to the synthesis of the numerical experiments and discusses the efficiency of our MIP approach.

## 2 A general 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 $S_{\text{dom}}$ of dominant job sequences, with respect to the feasibility problem, for the SMSP. Let us recall that a job sequence $\sigma_1$ dominates another job sequence $\sigma_2$ if $\sigma_2$ feasible $\Rightarrow$ $\sigma_1$ feasible. By extension, a set of job sequences $S_{\text{dom}}$ is said dominant if, for any job sequence $\sigma_2 \notin S_{\text{dom}}$, it exists $\sigma_1 \in S_{\text{dom}}$ such that $\sigma_2$ feasible $\Rightarrow \sigma_1$ 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 strictly include 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 $t_a$ 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 $m$ refers to as the total number of tops.

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

Considering the previous definition, we stress that a non-top job can belong to several pyramids. Let $u(j)$ ($v(j)$ resp.) refers to as 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 $S_{\text{dom}}$ of job sequences in the form:*

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

*where:*

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

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

## 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 that has been originally introduced in [4]. It aims at determining the most dominant job sequence among the set $S_{\text{dom}}$ in the form $\alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$. It is formulated below.

In the MIP, the binary variable $x^+_{ki}$ ($x^-_{ki}$ resp.) is set to 1 if the job $i$, assigned to Pyramid $P_k$, is sequenced in $\alpha_k$ (in $\beta_k$ resp.), provided that (see constraint (3.7)):

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

$$\text{s.t.} \quad \begin{cases} R_k \geq r_{t_k} & , \; \forall k \in [1\ m] & (3.1) \\ R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (3.2) \\ R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\ \qquad + \sum_{\{j \in P_k\}} p_j x^+_{kj} & , \; \forall k \in [2\ m] & (3.3) \\ D_k \leq d_{t_k} & , \; \forall k \in [1\ m] & (3.4) \\ D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (3.5) \\ D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\ \qquad - \sum_{\{j \in P_k\}} p_j x^-_{kj} & , \; \forall k \in [1\ (m-1)] & (3.6) \\ \sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) = 1 & , \; \forall i \in P_k & (3.7) \\ x^-_{ki} \ , \ x^+_{ki} \in \{0,1\} & , \; \forall k \in [1\ m], \forall i \in P_k & \\ D_k \ , \ R_k \in \mathbb{Z} & , \; \forall k \in [1\ m] & \end{cases}$$

- $u(i) \leq k \leq v(i)$;
- $i$ cannot be sequenced both in $\alpha_k$ and $\beta_k$;
- $i$ cannot be assigned to several pyramids.

The integer variable $R_k$ corresponds to the earliest starting time of Job $t_k$. By definition:

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

where $eft_{k-1}$ is the earliest completion time of the job subsequence $\beta_{k-1}$. 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:

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

where $lst_{k+1}$ is the latest starting time of the job subsequence $\alpha_{k+1}$. 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 $x^+_{ki}$ and $x^-_{ki}$ binary variables. In [4], it is shown that if $z = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k})$ is maximized

while respecting the constraints, then the obtained sequence dominates all the others. Indeed, the authors give the proof that, for any feasible combination of the $x^+_{ki}$ and $x^-_{ki}$ variables respecting the MIP constraints, a job sequence is obtained having its maximum lateness $L_{\max}$ strictly equals to $-z$. Therefore, maximizing $z$ is strictly equivalent to minimize 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.

## 4 Dominance condition for the $1|r_j|\sum U_j$ problem

In this section, the $\sum U_j$ criterion is considered. Searching 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 $\sum_{k=1\ldots n} C^k_n$ 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. □

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 if a job sequence belongs to the set of dominant sequences that Theorem 1 characterizes. For building up a master-pyramid-sequence associated with a job selection $E \subseteq V$, the $m_E$ tops and pyramids have first to be determined. Then, knowing that the set of dominant sequences is in the form $\alpha_1(E) \prec t_1(E) \prec \beta_1(E) \prec \cdots \prec \alpha_k(E) \prec t_k(E) \prec \beta_k(E) \prec \cdots \prec \alpha_{m_E}(E) \prec t_{m_E}(E) \prec \beta_{m_E}(E)$, it is assumed that any non-top job $j$ is sequenced both in $\alpha_k(E)$ and $\beta_k(E)$ (these subsequences being ordered as described in Theorem 1), $\forall k$ such that $u(j) \leq k \leq v(j)$. For illustration, let us consider a problem instance with 7 jobs such that the relative order among the release and due dates of the jobs is $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 < (r_5 = r_7) < d_6 < d_5 < d_1 < d_7$. For this example, the-master-pyramid-sequence associated with the selection $E = V$ is (tops are in bold):

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

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 $s \Subset \sigma_\Delta(V)$. Under the hypothesis that all tops are on-time, it is obvious that $\sigma_\Delta(V)$ also characterizes the set of dominant sequences (according to the $\sum U_j$ criterion) of any job selection $E$ such that $\{t_1, \ldots, t_m\} \subseteq E$. Indeed, the master-pyramid sequence $\sigma_\Delta(E)$ associated with such a selection is necessarily compatible with the master-pyramid sequence $\sigma_\Delta(V)$, *i.e.*, 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 total order: $r_d < r_b < r_c < r_a < d_a < d_b < d_c < d_d$. Job $a$ is the top of the corresponding structure and the master-pyramid sequence $\sigma_\Delta(V)$ is $(d, b, c, \mathbf{a}, b, c, d)$. Now, let us imagine that $a$ is not selected (it is late and its interval can be ignored). In this case, there are two tops $b$ and $c$ and the master-pyramid sequence $\sigma_\Delta(V \setminus \{a\})$ is $(d, \mathbf{b}, d, \mathbf{c}, d)$. As it can be observed, $\sigma_\Delta(V \setminus \{a\})$ is not compatible with $\sigma_\Delta(V)$ since, in the former, Job $d$ cannot be sequenced between $b$ and $c$, while it is possible in the latter. 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.

From now the focus is on a particular SMSP where any pyramid $P_k$, $\forall k = 1, \ldots, m$ is said *perfect*, *i.e.*, $\forall (i, j) \in P_k \times P_k$, $(r_i \geq r_j) \Leftrightarrow (d_i \leq d_j)$, *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 2** *Given a perfect SMSP $V$, the master-pyramid sequence $\sigma_\Delta(V)$ characterizes the complete set of sequences being dominant 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 $\alpha_k$, $\beta_k$ 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 $x$ (*i.e.*, $\sigma_\Delta(V) = (\alpha_1, t_1, \ldots, t_{x-1}, \beta_{x-1}, \alpha_x, j, \beta_x, \alpha_{x+1}, t_{x+1}, \ldots, t_m, \beta_m)$). 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 $\alpha_x$ (*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 $\alpha_x$ matches the reverse

order of the jobs of $\beta_x$, therefore $k$ is also the first job of $\beta_x$ (*i.e.*, $\beta_x = (k, \beta'_x)$). Then $\sigma_\Delta(V \setminus \{j\}) = (\alpha_1, t_1 \dots t_{x-1}, \beta_{x-1}, \alpha'_x, k, \beta'_x, \alpha_{x+1}, t_{x+1} \dots t_m, \beta_m)$ and in this case, $\sigma_\Delta(V \setminus \{j\})$ is obviously compatible with $\sigma_\Delta(V)$. □

## 5 Computing an upper bound for the $1|r_j|\sum U_j$ problem

In this section, we take an interest in finding a feasible job sequence in the form $\alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$ that minimizes the $\sum U_j$ criterion, knowing that a non-top $j$ is not necessarily sequenced, that is the major difference with Section 3. As discussed in Section 4, the sequences in this form are dominant for the $\sum U_j$ criterion only under the hypothesis that the top jobs are scheduled on time. As it does not necessarily exist any optimal sequence satisfying this assumption, finding an optimal sequence in this desired form only gives an upper bound to the general $1|r_j|\sum U_j$ problem. Nevertheless, as shown by the experiments (see Section 7), this upper bound is good and even often optimal.

The MIP formulation of Section 3 can easily be adapted for solving the previous problem:

$$\min \; z = \sum_{\{j \in V \setminus \{t_1 \dots t_m\}\}} (1 - \sum_{k=u(j)}^{v(j)} (x^-_{jk} + x^+_{jk})) + \sum_{k=1}^{m} y_{t_k}$$

$$\text{s.t.} \begin{cases}
R_k \geq r_{t_k} & , \; \forall k \in [1\ m] & (5.1)\\
R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (5.2)\\
R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j \in P_k\}} p_j x^+_{kj} & , \; \forall k \in [2\ m] & (5.3)\\
D_k \leq d_{t_k} & , \; \forall k \in [1\ m] & (5.4)\\
D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (5.5)\\
D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j \in P_k\}} p_j x^-_{kj} & , \; \forall k \in [1\ (m-1)] & (5.6)\\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) \leq 1 & , \; \forall i \in P_k & (5.7)\\
D_k - R_k \geq p_{t_k}(1 - y_{t_k}) & , \; \forall k \in [1\ m] & (5.8)\\
y_{t_k} \;,\; x^-_{ki} \;,\; x^+_{ki} \in \{0,1\} & , \; \forall k \in [1\ m], \forall i \in P_k & \\
D_k \;,\; R_k \in \mathbb{Z} & , \; \forall k \in [1\ m] &
\end{cases}$$

Let us comment this MIP. First, constraints (5.1)-(5.6) are identical to constraints (3.1)-(3.6) since integer variables $R_k$ and $D_k$ are determined in the same way. Allowing a non-top job to be late is easy by relaxing constraint (3.7), replacing it by constraint (5.7). As the feasibility of the obtained sequence is required, the constraint $D_k - R_k \geq p_{t_k}$ is set, $\forall k = 1, \dots, m$. Nevertheless, we observe that this constraint is a bit 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 $y_{t_k}$ is introduced (see constraint

(5.8)): $y_{t_k}$ equals 1 if the processing time of $t_k$ is ignored, 0 otherwise. Therefore, the $\sum U_j$ criterion can easily be expressed using the binary variables $y_{t_k}$, $x^+_{ki}$ and $x^-_{ki}$ since, if $y_{t_k} = 1$, Top $t_k$ is late and, if $\sum_{k=u(j)}^{v(j)}(x^-_{jk} + x^+_{jk}) = 0$, non-top job $j$ is late.

## 6 Lower-bound for the $1|r_j|\sum U_j$ problem

In a minimization problem, a lower bound is obtained by relaxing some constraints and optimally solving the relaxed problem. According to Theorem 2, when all the pyramids are perfect, 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. Moreover, for any problem, we know that it is always possible to decrease the $r_j$ values (or increase the $d_j$ values) of some jobs in order to make the pyramids perfect, *i.e.*, such that $\forall (i,j) \in P_k \times P_k$, $(r_i \geq r_j) \Leftrightarrow (d_i \leq d_j)$, $\forall k = 1, \ldots, m$ (see Figure 1). Doing so, a relaxed problem is obtained that can be optimally solved by the following MIP:

$$\min \; z = \sum_{\{j \in V \setminus \{t_1 \ldots t_m\}\}} (1 - \sum_{k=u(j)}^{v(j)} (x^-_{jk} + x^+_{jk})) + \sum_{k=1}^{m} y_{t_k}$$

$$\text{s.t.} \quad \begin{cases}
R_k \geq r_{t_k} + y_{t_k}(r_{n_k} - r_{t_k}) & , \; \forall k \in [1\ m] & (6.1) \\
R_k \geq r_i + (1 - x^+_{ik})(r_{n_k} - r_i) & & \\
\qquad + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (6.2) \\
R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j \in P_k\}} p_j x^+_{kj} & , \; \forall k \in [2\ m] & (6.3) \\
D_k \leq d_{t_k} + y_{t_k}(d_{n_k} - d_{t_k}) & , \; \forall k \in [1\ m] & (6.4) \\
D_k \leq d_i + (1 - x^-_{ik})(d_{n_k} - d_i) & & \\
\qquad - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \; \forall k \in [1\ m], \forall i \in P_k & (6.5) \\
D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j \in P_k\}} p_j x^-_{kj} & , \; \forall k \in [1\ (m-1)] & (6.6) \\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) \leq 1 & , \; \forall i \in P_k & (6.7) \\
D_k - R_k \geq p_{t_k}(1 - y_{t_k}) & , \; \forall k \in [1\ m] & (6.8) \\
y_{t_k} \;,\; x^-_{ki} \;,\; x^+_{ki} \in \{0,1\} & , \; \forall k \in [1\ m], \forall i \in P_k & \\
D_k \;,\; R_k \in \mathbb{Z} & , \; \forall k \in [1\ m] &
\end{cases}$$

with:

- $r_{n_k} = \min_{\{j \in P_k\}} r_j$;
- $d_{n_k} = \max_{\{j \in P_k\}} d_j$;

This MIP differs from the one of Section 5 only by the addition of the terms in bold that allow to deactivate the constraints of type (6.1), (6.2), (6.4) or

(6.5) when some jobs are late. For instance, if $t_k$ is late, *i.e.*, $y_{t_k} = 1$, the term $y_{t_k}(r_{n_k} - r_{t_k})$ ($y_{t_k}(d_{n_k} - d_{t_k})$ resp.) deactivates the constraint (6.1) (the constraint (6.4) resp.). Indeed, we know that the inequality $R_k \geq r_{n_k}$ (resp. $D_k \leq d_{n_k}$) is obviously always verified. Similarly, in the case where $i \notin \alpha_k$ ($i \notin \beta_k$ resp.), the term $(1 - x_{ik}^{+})(r_{n_k} - r_i)$ ($(1 - x_{ik}^{-})(d_{n_k} - d_i)$ resp.) deactivates the constraint (6.2) (the constraint (6.5) resp.). Note that the deactivation of constraints allows to ensure that only the constraints that concern the on-time jobs are taken into account.

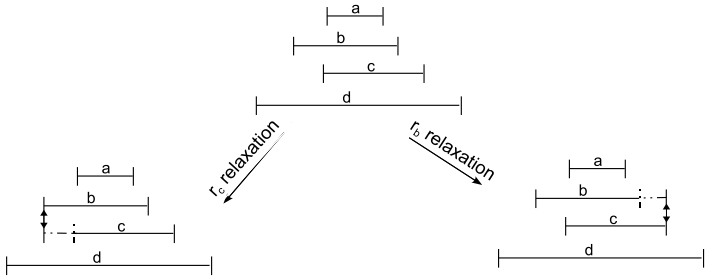


**Fig. 1.** Two relaxation strategies for turning a problem into a perfect one

## 7 Numerical experiments

For evaluating the performances of our MIP models, Baptiste et al.'s problem instances have been used (see [2]). For $n \in \{80, 100, 120, 140\}$, 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:

- two lower bounds by solving the MIP corresponding to the perfect SMSP obtained either by relaxation of the $r_i$ ($LBr_i$) or by relaxation of the $d_i$ ($LBd_i$), as explained in Section 6 ;
- one upper bound ($UB$) by solving the MIP described in Section 5.

In each experiment, the CPU time has been bounded to one hour. Table 1 displays, for the three kinds of MIP, the percentages of instances that were optimally solved within one hour, as well as the min / mean / max CPU time. For instance, when $n = 80$, the solver returns the optimal $LBr_i$ value in 94.16% of the cases, with a min / mean / max CPU time of 0.01/42.35/1395.4 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).

| *Instances* | $LB_r$ N/$Tcpu$ | $LB_d$ N/$Tcpu$ | UB N/$Tcpu$ |
|---|---|---|---|
| $n$=80 | 94.16 % (0.01; 42.35; 1395.54)s | 96.66 % (0.02;61.15; 2363.76)s | 98.33 % (0.02; 27.03; 1757.48)s |
| $n$=100 | 82.50 % (0.02; 141.15; 3531.11)s | 81.66 % (0.03; 85.70; 1318,86)s | 94.13 % (0.02; 33.84; 1778.42)s |
| $n$=120 | 80.83 % (0.02; 106.78; 1340.29)s | 84.16 % (0.04; 108.43; 2149.83)s | 85 % (0.02; 127.67; 2600.95)s |
| $n$=140 | 65.83 % (0.05; 139.77; 1490.64)s | 65.00 % (0.03; 173.97; 3072.82)s | 73.33 % (0.02; 134.62; 2600.95)s |

**Table 1.** Percentage of MIP solved to optimality

A few observations can be made at this point. First, even if some problem instances seem very hard to solve, optimal solutions are found in most of the cases. The computation of the upper bound is globally less time expensive than the one of the lower bound. This is not surprising since in the former case, because tops are assumed to be on time, the search space is less extended than in the latter case, where any job can be late or on time. We also observe in our experiments that the lower bound given by the relaxation of the $d_i$ is often better than the one obtained by relaxation of the $r_i$. Since the two kinds of relaxation are symetric, this is possibly due to the way problem instances have been generated.

| | $UB = LB$ with $LB = LB_r = LB_d$ | | $\delta = UB - LB$ |
|---|---|---|---|
| Instances | All instances | Instances: $Tcpu < 1h$ | $(min, mean, max)$ |
| $n$=80 | 70.83 % | 73.21% | (0; 0.38; 6) |
| $n$=100 | 70% | 73.62 % | (0; 0.39; 2) |
| $n$= 120 | 56.66% | 60.68 % | (0; 0.5; 2) |
| $n$=140 | 60.83 % | 62.31 % | (0; 1.45; 2) |

**Table 2.** Percentages of optimal solutions

Table 2 takes an interest in the cases where the solution is optimal, *i.e.*, the upper bound equals one of the lower bounds ($UB = LB = \max(LBr_i, LBd_i)$. Two sub-cases are distinguished according if one considers the total set of instances or only the instances for which the CPU time is lower than one hour. As one can see, optimality can be proved in many cases, even if the Baptiste et al.'s ad-hoc approach remains better from this point of view. Let us point out that our MIP approach proves the optimality of 3 instances that were not optimally solved by Baptiste et al.. The table also indicates the min / mean / max difference between the upper bound and the best lower bound: it is always small even for the largest instances.

Lastly, Table 3 gives a more tightened analysis of the quality of our upper bound $UB$. It is compared with either the optimal value $OPT$ found by the

| *Instances* | Baptiste et al. | | Dauzère-Pérès et al. | | $UB = OPT$ $Tcpu < 1h$ | | $UB \leq BEST$ |
|---|---|---|---|---|---|---|---|
| $n$=80 | 96.70 % | 117.3 s | 98.3 % | 49.0 s | 95.83 % | 27.03 s | 100 % |
| $n$=100 | 90.00 % | 273.5 s | 95.0 % | 78.4 s | 90.00 % | 33.84 s | 100 % |
| $n$=120 | 84.20 % | 538.2 s | 93.3 % | 89.70 s | 81.66 % | 127.64 s | 99.17% |
| $n$=140 | 72.50 % | 1037.3 s | 73.3 % | 233 s | 71.66 % | 134.62 s | 98.33 % |

**Table 3.** Analysis of the upper bound quality

Baptiste et al.'s method (when it is computed in less than one hour) or with the best solution $BEST$ that was returned otherwise. We observe that, nearly for all the instances, our upper-bound equals the optimal or the best solution found by the Baptiste et al's method. Moreover, we also observe that its computation is less time expensive in any 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, for instance intending to minimize the sum of their variations, could be explored.

## 8 Conclusion

Designing MIP models for solving efficiently basic SMSPs is of interest since MIP approaches are often adaptable for 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), only 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), the paper gives a MIP model that allows to directly find the 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 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]), allow to definitively fix the value of some binary variables, namely those of $y_{t_k}$, $x^+_{ki}$ and $x^-_{ki}$ in our MIP, intending to tighten the search space and speed up the solving phase. We guess that these techniques will improve our approach from a computational viewpoint such that it becomes more competitive in comparison with the best existing branch and bound approaches.

## 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. "Une formulation PLNE efficace pour $1|r_j|L_{\max}$", 10ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'09), Nancy (France), pp.104-105 (2009) (*in french*).
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 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*).