RAIRO-Oper. Res. 47 (2013) 33–46
DOI: 10.1051/ro/2013025

RAIRO Operations Research
www.rairo-ro.org

MINIMIZING THE NUMBER OF TARDY JOBS FOR THE SINGLE MACHINE SCHEDULING PROBLEM: MIP-BASED LOWER AND UPPER BOUNDS

Cyril Briand[1,2] and Samia Ourari[3]

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.

Mathematics Subject Classification. 90B35, 90C11.

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 job 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.

In this paper, we take an interest in finding a job sequence that minimizes the number of late jobs, problem referred as $1|r_j|\sum U_j$ in the literature, where U_j is

Received December 10, 2012. Accepted December 13, 2012.

[1] CNRS, LAAS, 7 avenue du colonel Roche, 31077 Toulouse, France. briand@laas.fr

[2] Université de Toulouse, UPS, INSA, INP, ISAE, LAAS, 31077 Toulouse, France

[3] CDTA, lotissement du 20 août 1956, Baba Hassen, Alger, Algeria. sourari@cdta.dz

set to 1 if job j is late, *i.e.*, $U_j \leftarrow (s_j + p_j > d_j)$. This problem has received considerable attention both in industry and in computer science where one has to decide whether to accept or to reject a new job, provided that all accepted jobs have to be completed on time on the machine (or the processor). This problem naturally also arises in more complex scheduling environment with several resources and precedence constraints, which gives motivation for determining procedures able to solve efficiently the single resource problem.

From the complexity viewpoint, let us mention that determining whether it exists a feasible sequence (*i.e.*, all jobs meet their due date) is NP-complete for the single machine problem [19]. The problem of minimizing the number of tardy jobs ($1|r_j|\sum U_j$) is strongly NP-hard [12], as well as the problem of minimizing the weighted number of tardy jobs ($1|r_j|\sum w_jU_j$). When $d_j = d$, the problem with weighted jobs ($1|d_j = d|\sum w_jU_j$) remains hard [14], whereas the unweighted case ($1|d_j = d|\sum U_j$) becomes polynomially solvable.

The paper is structured as follows. First, Section 2 proposes a brief analysis of the literature. Then Section 3 focuses on the problem of finding a feasible sequence to the SMSP and recalls a dominance theorem. In Section 4, a mathematical integer program (MIP) for determining a feasible job sequence, initially proposed in [6], is detailed. The following section discusses the validity of the dominance theorem stated in Section 3 for the case of the $\sum U_j$ criterion. Section 6 shows how the MIP of Section 4 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 presentation and the analysis of our computational experiments.

2. Related works

In the sequel, we review some important papers that directly deal with this problem, as well as some other approaches that consider additional assumptions or criteria.

For the basic NP-hard $1|r_j|\sum U_j$ and $1|r_j|\sum w_jU_j$ problems, efficient branch-and-bound procedures are reported in the literature [4,9,20]. The one of M'Hallah and Bulfin [20] is known as very effective since it is able to cope with both unweighted and weighted versions of the problem, solving instances with more than 200 jobs in less than 200 seconds. It also succeed in solving all problem instances that were reported as difficult by Baptiste *et al.* [4]. Many older works also deal with minimization of tardy jobs and we refer the reader to [20] for an up-to-date review.

When additional assumptions are made, $1|r_j|\sum U_j$ problems can become solvable in polynomial time. In the special case where release dates (or due dates) are equal, $1||\sum U_j$ problems can be solved in $\mathrm{O}(n\log(n))$ using Moore's well known algorithm [22]. Considering the case where release and due dates of jobs are agreeable, *i.e.*, $r_i < r_j \Rightarrow d_i \leq d_j$, Kise, Ibaraki and Mine proposed a dynamic programming algorithm that runs in $\mathrm{O}(n^2)$ [15]. Under this same assumption, an $\mathrm{O}(n\log(n))$ algorithm was later proposed by Lawler in 1982 [16]. Lawler [17] 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 $O(n^4)$ [3]. When processing times are equal, Carlier [7] proposed in the early eighties a $O(n^3\log(n))$ procedure. Nevertheless, this procedure has been proved incorrect by Chrobak *et al.* [8] who exhibit a new optimal $O(n^5)$ algorithm.

When jobs are weighted and release dates are identical, $1||\sum w_j U_j$ problems can be solved using dynamic programming or ad-hoc branch-and-bound procedures, as the efficient one of M'Hallah and Bulfin [21] that allows to consider problem instances with up to 2,500 jobs. Using variable fixing techniques, Baptiste *et al.* recently propose a MIP formulation [5] that solves a generalization of the previous problem with both due dates and deadlines, which is able to tackle problem instances with 50,000 jobs.

More recently, many works took an interest in some extensions of the $1|r_j|\sum U_j$ problem, taking other criterion or other constraints into consideration. In [25], the authors studied the problem of minimizing the number of tardy jobs when processing times are controllable. They consider a general convex decreasing resource consumption function, prove the NP-hardness of the problem, and assess the efficiency of three specific heuristics. Recently, focusing on the minimization of both flowtime and number of tardy jobs, Erenay *et al.* propose an algorithm based on a beam-search methodology and observe that it provides efficient schedules in most cases, comparing to other metaheuristic approaches [10] . Guohua and Benjamin [13] consider a single machine scheduling problem with dual criteria, *i.e.*, the minimization of the total weighted earliness subject to minimum number of tardy jobs, discuss several dominance properties of optimal solutions and propose efficient solving heuristic and exact procedures. Lee and Kim studied the $1|r_j|\sum U_j$ problem when periodic maintenance activities are imposed on the machine [18]. They provide a Mathematical Integer Program (MIP) to solve the problem optimally, propose a two-phase algorithm and give computational results for comparing both approaches in terms of quality. Tuon *et al.* consider a multi-agent version of the SMSP [24] where jobs are distributed among agents, each agent competing to perform his jobs on the machine. Every agent aims at minimizing the number of tardy jobs, provided a global objective function has also to be optimized. Considering both a $\varepsilon-$constraint approach and a linear combination of objectives, the authors give complexity results and provide a dynamic program that can tackle this multi-objective problem in pseudo-polynomial time. Aloulou and Della-Croce [2] consider a robust variant of the $1|r_j|\sum U_j$ problem where processing times are uncertain and modeled as a set of possible values (*i.e.*, scenarios). They take interest in determining the best worst-case among all the possible scenarios and give some complexity results. Aissi *et al.* also consider this problem when due dates are uncertain [1] and provide complexity results together with some approximation algorithms.

From this brief review of the literature, we observe that minimizing the number of tardy jobs is still an attractive objective due to its computational hardness on the one hand, and its practical interest, on the other hand. This paper describes a generic MIP for computing good-quality lower and upper bounds for the $1|r_j|\sum U_j$ problem. The MIP formulation is inspired by the one of Briand *et al.* [6] that solves the $1|r_j|L_{\max}$ efficiently. Determining good-quality bounds is of interest for boosting branch-and-bound methods or for providing good initial solution to local search procedures. Moreover, it is noteworthy that our MIP formulation can easily be adapted to tackle new constraints or new objectives.

3. A dominance theorem for the SMSP

This section focuses on the analysis of feasibility of job sequences for the SMSP with release and due dates. Some analytical dominance conditions are particularly recalled that have been originally introduced in the early eighties by Erschler *et al.* [11] within a theorem. This theorem uses the notions of a *top* and a *pyramid*, which are defined below. It characterizes a particular order between the tasks and exhibits a set S_{dom} of job sequences, said dominant with respect to feasibility of job sequences. Let us recall that a job sequence σ_1 dominates another job sequence σ_2 if σ_2 feasible $\Rightarrow \sigma_1$ 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 $\Rightarrow \sigma_1$ 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 3.1. 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 non-decreasing order with respect to their release dates or, in case of tie, in non-decreasing order with respect to their due dates. When both 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 3.2. 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 one of the top).

Considering the previous definition, let us 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 [11] for its proof.

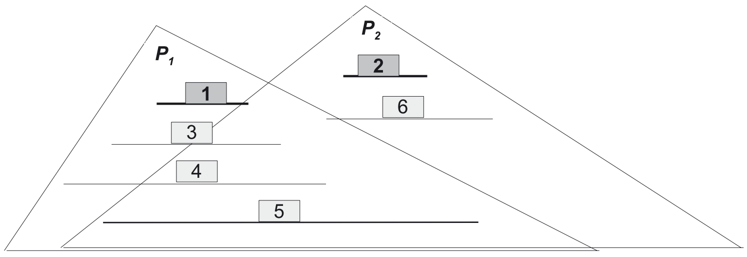

FIGURE 1. The interval structure, tops and pyramids.

Theorem 3.3. *A dominant set S_{dom} of job sequences is such that:*

- *the tops are ordered according to the non-decreasing 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 non-decreasing order of their release dates (in an arbitrary order in case of tie);*
- *only the jobs belonging to the last pyramid can be located after the last top and they are ordered according to the non-decreasing order of their due dates (in an arbitrary order in case of tie);*
- *between two tops t_k and t_{k+1}, only jobs belonging to P_k can be sequenced immediately after t_k, according to the non-decreasing order of their due dates (in an arbitrary order in case of tie), then jobs belonging to both P_k and P_{k+1} in an arbitrary order, and lastly jobs belonging only to P_{k+1} according to the non-decreasing order of their release dates (in an arbitrary order in case of tie).*

It should be pointed out that the numerical values of p_j, as well as the ones of r_j and d_j, are not used explicitly in the Theorem. Only the relative order of the release and due dates is considered, hence its interest. The number of characterized dominant sequences is: $card(S_{\text{dom}}) = \prod_{q=1}^{N}(q+1)^{n_q}$ where n_q is the number of non-top jobs belonging to exactly q pyramids and N is the total number of pyramids.

Moreover, let us highlight that any sequence belonging to S_{dom} is 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 α_k and β_k are job subsequences located at the left and the right of Top t_k respectively. Jobs assigned to α_k or β_k have to belong to P_k and they are sequenced with respect to the non-decreasing order of their r_j, for α_k, the non-decreasing order of their d_j, for β_k.

In order to illustrate the theorem, let us consider a problem instance with six jobs so that the relative order among the release dates r_j and the due dates d_j of the jobs is $r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < r_6 < d_4 < r_2 < d_2 < d_5 < d_6$. The interval structure associated with that example is displayed in Figure 1. There are two tops: $t_1 = 1$ and $t_2 = 2$ which characterize two pyramids: $P_1 = \{3, 4, 5\}$ and $P_2 = \{6\}$ (in accordance with the definition, a top job does not belong to the pyramid it characterizes).

According to Theorem 3.3, whatever the real values of r_i and d_i are (provided they remain compatible with the previous interval structure), the set S_{dom} of dominant sequences is in the form $\alpha_1 \prec 1 \prec \beta_1 \prec \alpha_2 \prec 2 \prec \beta_2$, where jobs belonging to subsequence α_k (β_k respectively) are sequenced with respect to the non-decreasing order of their r_j (d_j, respectively). Here we have $\alpha_1 \in \{4 \prec 5 \prec 3\}$, $\beta_1 \in \{3 \prec 4 \prec 5\}$, $\alpha_2 = \{6\}$, $\beta_2 \in \{5 \prec 6\}$, and $card(S_{\mathrm{dom}}) = (1+1)^3.(2+1)^1 = 24$ sequences (out of $6! = 720$ possible cases) are characterized:

$$\begin{array}{ll}
4 \prec 5 \prec 3 \prec 1 \prec 6 \prec 2\ , & 5 \prec 3 \prec 1 \prec 4 \prec 6 \prec 2 \\
4 \prec 5 \prec 3 \prec 1 \prec 2 \prec 6\ , & 5 \prec 3 \prec 1 \prec 4 \prec 2 \prec 6 \\
4 \prec 5 \prec 1 \prec 3 \prec 6 \prec 2\ , & 5 \prec 1 \prec 3 \prec 4 \prec 6 \prec 2 \\
4 \prec 5 \prec 1 \prec 3 \prec 2 \prec 6\ , & 5 \prec 1 \prec 3 \prec 4 \prec 2 \prec 6 \\
4 \prec 3 \prec 1 \prec 5 \prec 6 \prec 2\ , & 3 \prec 1 \prec 4 \prec 5 \prec 6 \prec 2 \\
4 \prec 3 \prec 1 \prec 5 \prec 2 \prec 6\ , & 3 \prec 1 \prec 4 \prec 5 \prec 2 \prec 6 \\
4 \prec 3 \prec 1 \prec 6 \prec 2 \prec 5\ , & 3 \prec 1 \prec 4 \prec 6 \prec 2 \prec 5 \\
4 \prec 3 \prec 1 \prec 2 \prec 5 \prec 6\ , & 3 \prec 1 \prec 4 \prec 2 \prec 5 \prec 6 \\
4 \prec 1 \prec 3 \prec 5 \prec 6 \prec 2\ , & 1 \prec 3 \prec 4 \prec 5 \prec 6 \prec 2 \\
4 \prec 1 \prec 3 \prec 5 \prec 2 \prec 6\ , & 1 \prec 3 \prec 4 \prec 5 \prec 2 \prec 6 \\
4 \prec 1 \prec 3 \prec 6 \prec 2 \prec 5\ , & 1 \prec 3 \prec 4 \prec 6 \prec 2 \prec 5 \\
4 \prec 1 \prec 3 \prec 2 \prec 5 \prec 6\ , & 1 \prec 3 \prec 4 \prec 2 \prec 5 \prec 6
\end{array}$$

4. A MIP for finding a feasible job sequence

In this section, the problem of finding a feasible job sequence is considered and a MIP is described, which has been originally introduced in [6]. 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 t_k, is strictly equivalent to the case where this job has been sequenced at the left of top t_{k+1}. These cases are not distinguished in the MIP model below.

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

$$\text{s.t.} \left\{ \begin{array}{lllll}
R_k & \geq & r_{t_k} & , \forall t_k \in \mathcal{T} & (4.1) \\
R_k & \geq & r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_{k-1,j} & , \forall i \in V \setminus \mathcal{T}, k = u(i) & (4.2) \\
R_k & \geq & R_{k-1} + p_{t_{k-1}} & & \\
 & + & \sum_{j \in P_{k-1}} p_j x_{k-1,j} & & \\
 & + & \sum_{\{j \in P_k | u(j) = k\}} p_j x_{k-1,j} & , \forall t_k \in \mathcal{T} \setminus \{t_1\} & (4.3) \\
D_k & \leq & d_{t_k} & , \forall t_k \in \mathcal{T} & (4.4) \\
D_k & \leq & d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_{k,j} & , \forall i \in V \setminus \mathcal{T}, k = v(i) & (4.5) \\
D_k & \leq & D_{k+1} - p_{t_{k+1}} & & \\
 & - & \sum_{\{j \in P_{k+1} | u(j) = k+1\}} p_j x_{kj} & & \\
 & - & \sum_{j \in P_k} p_j x_{kj} & , \forall t_k \in \mathcal{T} \setminus \{t_m\} & (4.6) \\
\multicolumn{3}{l}{\sum_{k=u(i)-1}^{v(i)} x_{ki} = 1} & , \forall i \in V \setminus \mathcal{T} & (4.7) \\
\multicolumn{3}{l}{x_{ki} \in \{0,1\}} & , \forall i \in V \setminus \mathcal{T}, \forall k \in [u(i)-1;\ v(i)] & \\
\multicolumn{3}{l}{D_k, R_k \in \mathbb{Z}} & , \forall t_k \in \mathcal{T} &
\end{array} \right.$$

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 4.7 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:

$$R_k = \max(r_{t_k}, eft_{k-1} + \sum_{\{j\in\alpha_k\}} p_j, \max_{i\in\alpha_k}(r_i + p_i + \sum_{\{j\in\alpha_k | i\prec j\}} p_j)) \tag{4.8}$$

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 (4.1), (4.2) and (4.3), according to Equation 4.8, 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:

$$D_k = \min(d_{t_k}, lst_{k+1} - \sum_{\{j\in\beta_k\}} p_j, \min_{i\in\beta_k}(d_i - p_i - \sum_{\{j\in\beta_k | j\prec i\}} p_j)) \tag{4.9}$$

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 (4.4), (4.5) and (4.6), according to Equation 4.9, give to D_k its value.

Obviously, it can be observed that the values of R_k and D_k variables can directly be deduced from the values of binary variables x_{ki}. In [6], the authors state 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 proof is given 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 $z = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k}) \geq 0$. 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 two tops t_1 and t_2 such that: $r_3 < r_1 < r_{t_1} < d_{t_1} < d_1 < d_3 < r_2 < r_{t_2} < d_{t_2} < d_2 < d_3$. There are two pyramids: $P_{t_1} = \{1, 3\}$ and $P_{t_2} = \{2, 3\}$. If the solution of the MIP is $x_{01} = 1, x_{11} = 0, x_{03} = 0, x_{13} = 1, x_{23} = 0$, $x_{12} = 0$ and $x_{22} = 1$ then the most dominant sequence is: $1 \prec t_1 \prec 3 \prec t_2 \prec 2$.

5. Dominance conditions for $1|r_j|\sum U_j$

In this section, the $\sum U_j$ criterion is considered again and some properties are stated. 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 3.3

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 5.1. *The union of all the dominant sequences that Theorem 3.3 characterizes for each possible selection of jobs is dominant for the $\sum U_j$ criterion. Note that, clearly, this property remains also valid with respect to $\sum w_j U_j$.*

Proof. The proof is obvious since the union of all the sequences that Theorem 3.3 characterizes for any possible selection necessarily includes the dominant sequences associated with E^*, hence an optimal solution. $\square$

As already pointed out, the number of possible job selections is quite large. Nevertheless, as explained in [23], 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 so-called *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 [9]. It allows to easily verify whether a job sequence belongs to a set of dominant sequences.The master-pyramid sequence associated with job selection $E \subseteq V$, with m_E tops and pyramids, is $\sigma_\Delta(E) = \alpha_1(E) \prec t_1(E) \prec \beta_1(E) \prec \cdots \prec \alpha_k(E) \prec t_k(E) \prec \beta_k(E) \prec \cdots \prec \alpha_{m_E}(E) \prec t_{m_E}(E) \prec \beta_{m_E}(E)$. According to Theorem 3.3, a non-top job j can be ranked inside subsequence $\alpha_k(E)$ with $k = u(j)$ or any subsequence $\beta_k(E)$ such that $k \in [u(j), v(j)]$.

For illustration, let us consider the problem instance with 6 jobs given in previous section. The master-pyramid sequence corresponding to the selection $E = V$ is (tops are in bold):

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

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 5.2. *Considering a given problem $1|r_j|\sum U_j$ with the set of jobs V, if an optimal solution exists where all tops are on-time 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 $s \Subset \sigma_\Delta(V)$. $\square$

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 shown in Figure 2. Job t is the single top of the interval structure and the master-pyramid sequence $\sigma_\Delta(V)$ is $(c \prec a \prec b) \prec \mathbf{t} \prec (a \prec b \prec c)$. 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,

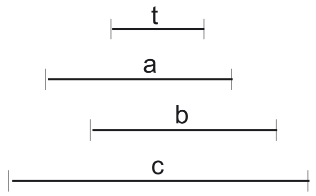

FIGURE 2. An interval structure.

the selection becomes $E = V \setminus \{t\} = \{a, b, c\}$ and there are two tops a and b that characterize the 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.

Up to now, we have shown that Theorem 3.3 is not dominant in general for the $\sum U_j$ criterion. Let us take interest now in a particular interval 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 5.3. *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 $\sigma_\Delta(V)$. $\square$

6. 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 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 non-top jobs to be late is easy by relaxing constraints (4.7), replacing them by constraints (5.7). As the feasibility of the obtained sequence is required, the constraints $D_k - R_k \geq p_{t_k}$ are 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 variables y_k are 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 (4.3) and (4.6) must be adapted (see constraints (5.3) and (5.6)) so that 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 (4.1) and (4.4) should be relaxed: this can be done by replacing them by constraints (5.1) and (5.4), 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 (4.2)–(4.5) 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 (5.5), respectively. Finally, the $\sum U_j$ criterion can easily be expressed by using binary variables y_k and x_{ki} since, when $\sum_{k=u(j)-1}^{v(j)} x_{jk} = 0$, the non-top job j is not assigned to any pyramid and can be considered late.

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

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

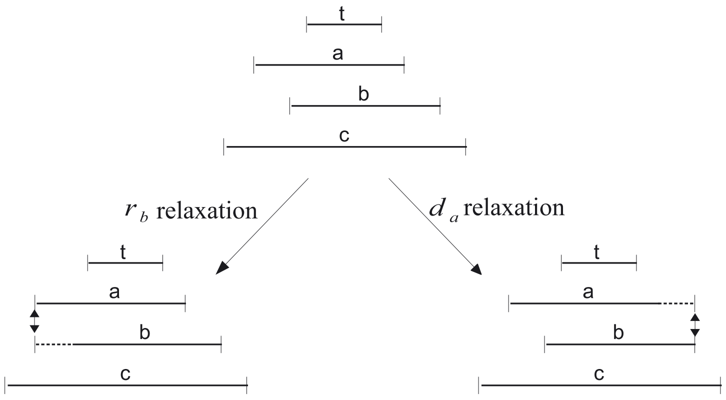

FIGURE 3. Two relaxation strategies for converting a problem into a perfect one.

Obviously, considering any problem V, solving the previous MIP gives an upper bound to the 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 5.3, 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 Fig. 3). 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.

7. Numerical experiments

For evaluating the performances of our MIP model, Baptiste *et al.*'s problem instances have been used (see [4]). 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

TABLE 1. Percentages of MIP solved to optimality.

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

TABLE 2. Percentages of optimal solutions.

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

due to the way problem instances have been generated (see [4]). 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.

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 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

TABLE 3. Analysis of the lower and upper bounds quality.

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

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 suggest that, in order to increase the percentage of optimal-certified solutions, our relaxation approach 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.

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 objectives. 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 Sect. 5), 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 [5]), might 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 such techniques could improve our approach from a computational viewpoint so that it hopefully becomes competitive with existing branch and bound procedures.

Acknowledgements. We would like to address a special thank to Philippe Baptiste who kindly provide us with his problem instances, as well as the detailed report of his experiments.

References

[1] H. Aissi, M.A. Aloulou and M.Y. Kovalyov, Minimizing the number of late jobs on a single machine under due date uncertainty. *J. Sched.* **14** (2011) 351–360.

[2] M.A. Aloulou and F. Della-Croce, Complexity of one machine scheduling problems under scenario-based uncertainty. *Oper. Res. Lett. 36* (2008) 338–342.

[3] P. Baptiste, Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal. *J. Sched.* **2** (1999) 245–252.

[4] P. Baptiste, L. Peridy and E. Pinson, A branch and bound to mininimze the number of late jobs on a single machine with release time constraints. *Eur. J. Oper. Res.* **144** (2003) 1–11.

[5] P. Baptiste, F. Della Croce, A. Grosso and V. T'kindt, Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances. *J. Sched.* **13** (2010) 39–47.

[6] C. Briand, S. Ourari and B. Bouzouia, An efficient ILP formulation for the single machine scheduling problem. *RAIRO Oper. Res.* **44** (2010) 61–71.

[7] J. Carlier, Problèmes d'ordonnancements à durées égales. *QUESTIO* **5**(4) (1981) 219–228.

[8] M. Chrobak, C. Dürr, W. Jawor, L. Kowalik and M. Kurowski, A Note on scheduling equal-length jobs to maximize throughput. *J. Sched.* **9** (2006) 71–73.

[9] S. Dauzère-Pérès and M. Sevaux, An exact method to minimize the number of tardy jobs in single machine scheduling. *J. Sched.* **7** (2004) 405–420.

[10] F.S. Erenay, I. Sabuncuoglu, A. Toptal and M.K. Tiwari, New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs. *Eur. J. Oper. Res.* **201** (2010) 89–98.

[11] J. Erschler, G. Fontan, C. Merce and F. Roubellat, A new dominance concept in scheduling n jobs on a single machine with ready times and due dates. *Oper. Res.* **31** (1983) 114–127.

[12] M.R. Garey and D.S. Johnson, Computers and intractability, a guide to the theory of NP-completeness. W. H. Freeman and Company (1979).

[13] W. Guohua and P.-C.Y. Benjamin, Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs. *Eur. J. Oper. Res.* **195** (2009) 89–97.

[14] R.M. Karp, Reducibility among combinatorial problems. in Complexity of Computer Computations, edited by R.E. Miller and J.W. Thatcher. Plenum Press, New York (1972) 85–103.

[15] H. Kise, I. Toshihide and H. Mine, A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.* **26** (1978) 121–126.

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

[17] 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** (1990) 125–133.

[18] J.Y. Lee, Y.D. Kim, Minimizing the number of tardy jobs in a single-machine scheduling problem with periodic maintenance. *Comput. Oper. Res.* **39** (2012) 2196–2205.

[19] J.K. Lenstra, A.H.G. Rinnooy Han and P. Brucker, Complexity of machine scheduling problems. *Ann. Discrete Math.* **1** (1977) 343–362.

[20] R. M'Hallah and R.L. Bulfin, Minimizing the weighted number of tardy jobs on a single machine with release dates. *Eur. J. Oper. Res.* **176** (2007) 727–744.

[21] R. M'Hallah, Bulfin, R.L., Minimizing the weighted number of tardy jobs on a single machine. *Eur. J. Oper. Res.* **145** (2003) 45–56.

[22] M.J. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Manag. Sci.* **15**(1) (1968) 102–109.

[23] S. Ourari and C. Briand 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) 351–352 (2008).

[24] N.H. Tuong, A. Soukhal and J.-C. Billaut, Single-machine multi-agent scheduling problems with a global objective function. *J. Sched.* **15** (2011) 311–321.

[25] L. Yedidsion, D. Shabtay, E. Korach and M. Kaspi, A bicriteria approach to minimize number of tardy jobs and resource consumption in scheduling a single machine. *Int. J. Product. Econom.* **119** (2009) 298–307.