AN EFFICIENT ILP FORMULATION FOR THE SINGLE MACHINE SCHEDULING PROBLEM

Cyril Briand^{1, 2}, Samia Ourari^{1, 2, 3} and Brahim Bouzouia³

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. On the basis of analytical and numerical dominance conditions, an efficient integer linear programming formulation is proposed for this problem, aiming at minimizing the maximum lateness (L_{max}) . Experiments have been performed by means of a commercial solver that show that this formulation is effective on large problem instances. A comparison with the branch-and-bound procedure of Carlier is provided.

Keywords. Single machine scheduling, integer linear programming, dominance conditions.

Mathematics Subject Classification. 90B35.

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 duedate. The processing time p_j of j is known and preemption is not allowed. A feasible job sequence has to be found so that, s_j being the starting time of job j, $s_j \geq r_j$ and $s_j + p_j \leq d_j$.

Received July 1, 2009. Accepted October 21, 2009.

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; sourari@cdta.dz

 $^{^3}$ CDTA, lotissement du 20 août 1956, Baba Hassen, Alger
, Algeria; ${\tt bbouzouia@cdta.dz}$

Seeking whether it exists a feasible sequence (i.e., all jobs meet their due dates) is NP-complete [6] and consequently, finding a sequence that minimizes the maximum lateness (problem refers to as $1|r_i|L_{\rm max}$) is NP-hard. For this problem, the well-known branch-and-bound procedure of Carlier [2] can be used for solving problems having more than one thousand jobs. Nevertheless, even if the Carlier's procedure provides good performance, it is still interesting to design other exact approaches that can be used in a more generic framework such as integer linear programming (ILP).

This paper presents an original ILP formulation for the SMSP that is based on analytical and numerical dominance conditions. The paper first presents a dominance theorem that is the foundation of the work. Then ILP models are progressively issued, starting from particular SMSP cases up to the general case. The last section is devoted to the experimentations.

2. A GENERAL DOMINANCE THEOREM FOR THE SMSP

In the sequel, some analytical dominance conditions are used for the SMSP. They have been originally proposed in the early eighties by Erschler *et al.* [4] within a theorem. This theorem uses the notions of a *top* and a *pyramid*, which are defined below. It defines a set S_{dom} of dominant job sequences for the SMSP with regard to the feasibility problem. Let us recall that a job sequence σ_1 dominates another job sequence σ_2 if σ_2 feasible $\Rightarrow \sigma_1$ is 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 $\Rightarrow \sigma_1$ is feasible. When searching a feasible job sequence, only the set of dominant sequences need to be explored since, if there does not exist any feasible sequence in the dominant set, then it can be asserted that the original problem does not admit any feasible solution. This leads to a significant reduction of the search space.

Definition 2.1. A job $t \in V$ is a top if there does not exist any other job $i \in V$ such that $r_i > r_t \wedge d_i < d_t$ (i.e., the execution window of a top job does not strictly include any other execution window).

The top jobs are indexed in ascending order with respect to their release dates or, in case of a 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 top jobs then a < b if and only if $(r_{t_a} \le r_{t_b}) \land (d_{t_a} \le d_{t_b})$. Let m be the total number of top jobs.

Definition 2.2. A pyramid P_k related to a top job t_k is the set of jobs $i \in V$ such that $r_i < r_{t_k} \land d_i > d_{t_k}$ (i.e., the set of jobs for which their execution window strictly includes the execution window of the top job).

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

The following theorem, proved in [4], can now be stated.

Theorem 2.1. The set of sequences in the form $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_k \prec t_k \prec \beta_k \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$ is dominant for finding a feasible sequence, where:

- t_k is the top job of the pyramid P_k , $\forall k = 1 \dots m$;
- α_k and β_k are two job subsequences located at the left and the right of top job t_k respectively, such that, for any k, the jobs of α_k are ordered by increasing release dates and the jobs of β_k are ordered by increasing due dates;
- any non-top job j is assigned to one sequence α_k or β_k , with $u(j) \leq k \leq v(j)$.

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 2.1, whatever the real values of r_i and d_i (provided they are 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, 24 dominant sequences (out of 7! = 5040) are characterized.

3. The mono-pyramidal SMSP

In this section, the focus is on the mono-pyramidal SMSP: it is assumed that the problem has a single top job $t \in V$ (hence a single pyramid P_t). With respect to Theorem 2.1, the set S_{dom} of dominant job sequences is in the form $\alpha \prec t \prec \beta$, where jobs belonging to subsequence α are sequenced with respect to the increasing order of their r_j , and jobs belonging to β , with respect to the increasing order of their d_j . Hence, 2^n sequences need to be explored for checking the feasibility (instead of (n+1)!). Below, another numerical dominance theorem is stated that allows to compare the 2^n sequences each other. It is based on a necessary and sufficient condition of feasibility.

As stated in [3] regarding the general SMSP, let us first recall that a job sequence σ is feasible iff:

$$p_i + \sum_{i \prec k \prec j} p_k + p_j \le d_j - r_i, \forall (i, j) \in \sigma \text{ such that } i \prec j.$$
 (3.1)

The previous necessary and sufficient condition of feasibility can be drastically simplified for the mono-pyramidal SMSP. For matter of notation simplification, given a dominant sequence in the form $\alpha \prec t \prec \beta$, the following notations are set:

- $\begin{array}{ll} \bullet & R_a = r_a + p_a + \sum_{k \in \alpha, a \prec k} (p_k), \forall a \in \alpha; \\ \bullet & D_b = d_b p_b \sum_{k \in \beta, k \prec b} (p_k), \forall b \in \beta; \\ \bullet & R^{\max} = \max[r_t, \max_{a \in \alpha} (R_a)]; \end{array}$

- $D^{\min} = \min [d_t, \min_{b \in \beta}(D_b)].$

Clearly, R^{max} corresponds to the earliest starting time of Top t, further refers to as est_t . Symmetrically, D^{\min} corresponds to the latest completion time of Top t, further refers to as lft_t .

Now the following theorem can be stated.

Theorem 3.1. P being a pyramid of top t, any dominant sequence in the form $\alpha \prec t \prec \beta$ is admissible if and only if $D^{\min} - R^{\max} - p_t \ge 0$.

Proof. Let us first prove that $D^{\min} - R^{\max} - p_t \ge 0$ implies the feasibility of $\alpha \prec t \prec \beta$, i.e., the condition (3.1) is verified. Condition (3.1) can be simplified since the case where i and j $(i \prec j)$ both belong to α can be ignored. Indeed, $D^{\min} - R^{\max} - p_t \ge 0$ implies that $d_t - R_i - p_t \ge 0 \ \forall i \in \alpha$ and, because $d_j > d_t$ $\forall j$, then $d_j - R_i - p_t > 0$. By replacing R_i by its value it comes that $d_j - r_i > 0$ $p_t + \sum_{j \prec k \prec t} p_k + p_j + \sum_{i \prec k \prec j} p_k + p_i > p_i + \sum_{i \prec k \prec j} p_k + p_j$. Therefore, if condition (3.1) holds between any job $i \in \alpha$ and the top job t, then it also holds between i and any $j \in \alpha$ such that $i \prec j$. A similar reasoning can be made for discarded the case where i and j both belongs to β . Indeed, if condition (3.1) holds between any $j \in \beta$ and the top job t, then it also hods between any $i \in \beta$ and j, provided that $i \prec j$. So we only need to prove that $D_b - R_a - p_t \geq 0$, $\forall a \in (\alpha \prec t) \text{ and } \forall b \in (t \prec \beta).$ This is obvious since $D_b - R_a \geq D^{\min} - R^{\max}$ by definition and $D^{\min} - R^{\max} - p_t \ge 0$ by assumption.

Now, we are going to prove that the feasibility of $\alpha \prec t \prec \beta$ implies that $D^{\min} - R^{\max} - p_t \ge 0$. It is also obvious since if $\alpha \prec t \prec \beta$ is feasible then, from condition (3.1), $D_b - R_a - p_t \ge 0$, $\forall (a, b) \in \alpha \times \beta$ and $d_t - r_t - p_t \ge 0$. Of course, this inequality needs to be verified in the worst case, i.e., $\min(d_t, \min_{b \in \beta} D_b)$ – $\max(r_t, \max_{a \in \alpha} R_a) - p_t \ge 0$. Hence $D^{\min} - R^{\max} - p_t \ge 0$.

The fact that Theorem 3.1 permits to numerically compare sequences each other is helpful since it immediately involves the following dominance condition.

Corollary 3.1. P being a pyramid, job t being its top job, and σ_1 and σ_2 being two dominant sequences in the form $\alpha_1 \prec t \prec \beta_1$ and $\alpha_2 \prec t \prec \beta_2$ respectively, if $D_1^{\min} - R_1^{\max} \ge D_2^{\min} - R_2^{\max}$, then σ_1 dominates σ_2 with regard to the feasibility

Proof. By definition, stating that σ_1 dominates σ_2 requires to prove that the feasibility of σ_2 implies the one of σ_1 . Now we know that σ_2 is feasible iff (see Thm. 3.1) $D_2^{\min} - R_2^{\max} \ge p_t$, therefore, under the assumption that $D_1^{\min} - R_1^{\max} \ge D_2^{\min} - R_2^{\max}$, it can be deduced that $D_1^{\min} - R_1^{\max} \ge p_t$, so σ_1 is necessarily also feasible.

Consequently, the problem of searching a feasible job sequence in a monopyramidal SMSP can be reformulated as an optimization problem where the objective function is to maximize $D^{\min} - R^{\max}$. An integer linear program (ILP1) is given below for modelling such a problem.

$$\text{s.t.} \begin{cases} R \geq r_t & (3.2) \\ D \leq d_t & (3.3) \\ R \geq r_i + \sum_{\{j \in V \setminus \{t\} \mid r_j \geq r_i\}} p_j x_j^+, \ \forall i \in V \quad (3.4) \\ D \leq d_i - \sum_{\{j \in V \setminus \{t\} \mid d_j \leq d_i\}} p_j x_j^-, \ \forall i \in V \quad (3.5) \\ x_i^- + x_i^+ = 1 & , \ \forall i \in V \quad (3.6) \\ x_i^-, x_i^+ \in \{0,1\} & , \ \forall i \in V \\ D, R \in \mathbb{Z}. \end{cases}$$

In the above ILP1 formulation, every job $j \in V \setminus \{t\}$ has to be sequenced either at the left side $(i.e., x_j^+ = 1 \text{ and } j \in \alpha)$ or at the right side $(i.e., x_j^- = 1 \text{ and } j \in \beta)$ of Top t, but not both $(i.e., x_i^- + x_i^+ = 1)$. If z = D - R is maximized, while respecting the constraints, then, from Corollary 3.1, the obtained sequence dominates all the others. Moreover, if $z^* \geq p_t$ then, from Theorem 3.1, the sequence is feasible. If $z^* < p_t$ then it can be asserted that it does not exist any feasible sequence for the considered problem.

We highlight that, since $x_i^- + x_i^+ = 1$, the variables x_i^- (or x_i^+) are redundant and can be removed from the ILP. Nevertheless, we prefer to keep them for enhancing the clarity of the model.

Another important remark concerns the constraints (3.4). According to Theorem 3.1, they only need to be satisfied for the jobs of α . What happens now if a job x is in β ? Could the constraint of type (3.4) related to x be unnecessarily restrictive? We prove below that, in such a case, this constraint cannot be active since it always exists a job $y \in (\alpha \prec t)$ inducing a more restrictive constraint, *i.e.*:

$$r_y + \sum_{\{j \in V \setminus \{t\} | r_j \ge r_y\}} p_j x_j^+ \ge r_x + \sum_{\{j \in V \setminus \{t\} | r_j \ge r_x\}} p_j x_j^+.$$

Indeed, let y be the job in $(\alpha \prec t)$ having the lowest release date r_y , provided that $r_y \geq r_x$. Since we obviously have $\sum_{\{j \in V \setminus \{t\} \mid r_j \geq r_y\}} p_j x_j^+ = \sum_{\{j \in V \setminus \{t\} \mid r_j \geq r_x\}} p_j x_j^+$, the previous relation necessarily holds.

A similar reasoning can be made for the constraints (3.5) since, if a job x is in α , then, considering the job $y \in (t \prec \beta)$ having the greatest due date d_y , provided that $d_y \leq d_x$, on can proved that:

$$d_y - \sum_{\{j \in V \setminus \{t\} | d_j \le d_y\}} p_j x_j^- \le d_x - \sum_{\{j \in V \setminus \{t\} | d_j \le d_x\}} p_j x_j^-.$$

Now, let us show that the optimal solution of ILP1 is also optimal with regard to the maximum lateness minimization.

Theorem 3.2. The optimal ILP1 solution minimizes the maximum lateness and $L_{\text{max}}^* = p_t - z^*$.

Proof. Let $\alpha \prec t \prec \beta$ be a feasible solution of ILP1 and z=D-R be the corresponding objective value. Let us determine the lateness of a job $a \in \alpha$. By definition, when all the jobs are scheduled at the latest, the maximum lateness L_a of a equals $L_a=r_a-lst_a$, where lst_a refers to as the latest starting time of a. Obviously, there exists a job $u \in \beta$ such that $D=d_u-\sum_{\{k\in\beta|d_k\leq d_u\}}p_k$. Therefore, the latest starting time of t is $lst_t=D-p_t$ and the latest starting time of any job $a\in\alpha$ is $lst_a=D-p_t-\sum_{\{k\in\alpha|a\prec k\}}p_k-p_a$. Then it is easy to see that the job $v\in\alpha$, which has the maximum lateness $(i.e., L_v=\max_{a\in\alpha}(L_a))$ is precisely the one that gives to R its value. Hence $L_v=R-D+p_t$.

Let us now determine the lateness L_b of a job $b \in \beta$. Symmetrically to the previous reasoning, one can determine that the earliest finishing time of b is $eft_b = R + p_t + \sum_{\{k \in \beta \mid k \prec b\}} p_k + p_b$. Since, when all the jobs are scheduled at the earliest, $L_b = eft_b - d_b$, the job $u \in \beta$ which has the maximum lateness $(i.e., L_u = \max_{b \in \beta}(L_b))$ is precisely the one that gives to D its value. Hence $L_u = R - D + p_t$. In conclusion, there is a strict equivalence between the maximum lateness minimization and the maximization of the objective function z since $L_{\max} = p_t - z$.

4. The independent multi-pyramidal SMSP

Before considering the general SMSP, let us focus on the particular SMSP characterized by m top jobs, hence m pyramids, such that any non-top job only belongs to a single pyramid (*i.e.*, the pyramids are independent). With respect to Theorem 2.1, any dominant job sequence is in the form $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \ldots \prec \alpha_m \prec t_m \prec \beta_m$, where t_k is the top job of the pyramid P_k , $\forall k = 1 \ldots m$ and the jobs of α_k and β_k only belong to pyramid P_k .

Therefore, the problem of finding the most dominant job sequence for the independent multi-pyramidal SMSP can be decomposed into m interdependent mono-pyramidal subproblems. For each of them, the most dominant job sequence $\alpha_k \prec t_k \prec \beta_k$ has to be found. In other words, if R_k refers to as the earliest starting time of Top t_k and D_k to as the latest completion time of Top t_k , we have to maximize $D_k - R_k - p_{t_k}$.

These mono-pyramidal subproblems are interdependent because the sequences $\alpha_k \prec t_k \prec \beta_k$ should not overlap. This requirement can easily be achieved by setting, for any job $j \in P_k$, the constraints: $r_j = \max(r_j, eft_{k-1})$ and $d_j = \min(d_j, lst_{k+1})$, where eft_{k-1} (lst_{k+1} resp.) refers to as the earliest completion time of Subsequence β_{k-1} (the latest starting time of Subsequence α_{k+1} resp.).

The value of eft_{k-1} corresponds to the earliest completion time of Top t_{k-1} (i.e., $R_{k-1} + p_{t_{k-1}}$), plus the sum of the processing times of the jobs of β_{k-1} ,

i.e. $eft_{k-1} = R_{k-1} + p_{t_{k-1}} + \sum_{\{j \in P_{k-1}\}} (p_j x_j^-)$. Similarly, the value of lst_{k+1} corresponds to the latest starting time of Top t_{k+1} (i.e. $D_{k+1} - p_{t_{k+1}}$), minus the sum of the processing times of the jobs of α_{k+1} , i.e. $lst_{k+1} = D_{k+1} - p_{t_{k+1}} - \sum_{\{j \in P_{k+1}\}} (p_j x_j^+)$. Therefore, eft_{k-1} and lst_{k+1} only depend on the ILP variables. Consequently, the problem of searching a feasible job sequence in a multipyramidal independent SMSP can be formulated by the following ILP (ILP2):

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

$$\begin{cases} R_k \ge r_{t_k} &, \forall k \in [1 \ m] \\ D_k \le d_{t_k} &, \forall k \in [1 \ m] \end{cases} & (4.1) \\ R_k \ge r_i &+ \sum_{\{j \in P_k \mid r_j \ge r_i\}} p_j x_j^+ &, \forall i \in P_k \\ D_k \le d_i &- \sum_{\{j \in P_k \mid d_j \le d_i\}} p_j x_j^- &, \forall i \in P_k \\ R_k \ge R_{k-1} &+ \sum_{\{j \in P_{k-1}\}} p_j x_j^- &+ p_{t_{k-1}} \\ &+ \sum_{\{j \in P_k \mid r_j \ge r_i\}} p_j x_j^+ &, \forall k \in [2 \ m] \\ D_k \le D_{k+1} &- \sum_{\{j \in P_{k+1}\}} p_j x_j^+ &- p_{t_{k+1}} \\ &- \sum_{\{j \in P_k \mid d_j \le d_i\}} p_j x_j^- &, \forall k \in [1 \ (m-1)] \\ x_i^- &+ x_i^+ &= 1 &, \forall i \in P_k \\ T_k^- &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ \\ T_k^- &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ \\ T_k^- &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ \\ T_k^- &+ T_k^+ &+ T_k^+ &+ T_k^+ &+ T_k^+ \\ T_k^- &+ T_k^+ \\ T_$$

In the previous formulation, similarly with ILP1, the binary variable x_i^+ (x_i^- resp.) is set to 1 if Job i, which only belongs to pyramid P_k , is sequenced in α_k (in β_k resp.), provided that it cannot be sequenced both in α_k and β_k (i.e., $x_i^- + x_i^+ = 1$). The constraints (4.1)–(4.4) are identical with the constraints (3.2)–(3.5) of ILP1. The constraints (4.5) and (4.6) impose that the subsequences $\alpha_k \prec t_k \prec \beta_k$ do not overlap. Under these constraints, one can determine for each pyramid P_k , the maximum lateness, which equals $R_k - D_k + p_{t_k}$. Hence, maximizing z is equivalent to minimize the global maximum lateness.

5. The general SMSP

Now, the general SMSP is considered. Here, a non-top job j can belong to several pyramids, that is the only difference with the multi-pyramidal independent SMSP. Let us note u(j) (v(j) resp.) the index of the first pyramid (the last pyramid resp.) to which the job j belongs. With respect to Theorem 2.1, any dominant sequence is in the form $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$, where t_k is the top job of the pyramid P_k , $\forall k = 1 \ldots m$, and any non-top job j can be sequenced either in α_k or β_k , for any k such that $u(j) \leq k \leq v(j)$.

It is easy to see, once the assignments of the non-top jobs to their pyramid done, that the problem turns into a multi-pyramidal independent problem (that can be solved using the ILP2 formulation). Consequently, a general formulation can be obtained by simply adding binary variables to the ILP2 formulation, so that each non-top job can only be assigned to a single pyramid P_k . The following general formulation (ILP3) is then obtained:

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

$$\begin{cases} R_k \ge r_{t_k} &, \ \forall k \in [1 \ m] \\ D_k \le d_{t_k} &, \ \forall k \in [1 \ m] \end{cases} & (5.1) \\ R_k \ge r_i &+ \sum_{\{j \in P_k | r_j \ge r_i\}} p_j x_{kj}^+ &, \ \forall i | k = u(i) \end{cases} & (5.2)$$

$$R_k \le d_i &- \sum_{\{j \in P_k | d_j \le d_i\}} p_j x_{kj}^- &, \ \forall i | k = v(i) \end{cases} & (5.4)$$

$$R_k \ge R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x_{(k-1)j}^+ &+ p_{t_{k-1}} \\ &+ \sum_{\{j \in P_k | r_j \ge r_i\}} p_j x_{kj}^+ &, \ \forall k \in [2 \ m] \end{cases} & (5.5)$$

$$D_k \le D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x_{(k+1)j}^+ &- p_{t_{k+1}} \\ &- \sum_{\{j \in P_k | d_j \le d_i\}} p_j x_{kj}^- &, \ \forall k \in [1 \ (m-1)] \end{cases} & (5.6)$$

$$\sum_{k=u(i)}^{v(i)} (x_{ki}^- + x_{ki}^+) = 1 &, \ \forall i \end{cases} & (5.7)$$

$$x_{ki}^- &, x_{ki}^+ &\in \{0,1\} \\ D_k &, R_k \in \mathbb{Z} &, \ \forall k \in [1 \ m].$$
ILP3, similarly with ILP2, the binary variable $x_{ki}^+ (x_{ki}^- \text{ resp.})$ is set to 1 if Job

In ILP3, similarly with ILP2, the binary variable x_{ki}^+ (x_{ki}^- resp.) is set to 1 if Job i, belonging to Pyramid P_k (i.e. $u(i) \le k \le v(i)$), is sequenced in α_k (in β_k resp.), provided that it cannot be sequenced both in α_k and β_k and it is assigned to a single pyramid (i.e., the constraint (5.7) is satisfied for i). The constraints (5.1)–(5.6) remain unchanged, when compared to ILP2.

Now let us give few comments concerning the number of constraints and variables that are involved. There is one constraint of type (5.1) and (5.2) for each top job. There is one constraint of type (5.3) and (5.4) for each non-top job and there are (n-m) non-top jobs. There are (m-1) constraints of type (5.5) and (m-1) constraints of type (5.6). There is one constraint of type (5.7) for each non-top job. Therefore, the total number of constraint is (m+3n-2). The number of binary variables is $(2\sum_{k=1}^{m}(n_k))$, where n_k is the number of non-top jobs that can be assigned to P_k . There are (2m) integer variables but we notice that their value can be directly deduced from those of the binary variables.

6. Numerical experiments

The ILP3 model has been implemented using the Ilog-concert library and tested on problem instances having from 100 up to 3100 jobs. For each problem instance, we both ran the ILP3 model and the Carlier's procedure, aiming at comparing

	All instances								Hard instances						
n	$T_{\it Car}$			$T_{I\!L\!P}$			%W	# <i>S</i>	$T_{\it Car}^{\it hard}$			$T_{I\!LP}^{hard}$			% W ^{hard}
	min	mean	max	min	mean	max			min	mean	max	min	mean	max	
100	0	0.02	1	0	0.04	1	4	16	0	0.06	1	0	0.06	1	2.9
400	0	0.3	2	0	0.22	1	16	38	0	1	2	0	0.33	1	8.3
700	0	1.4	6	0	1.24	8	40	35	1	3.66	6	0	1.33	3	0
1000	0	1.22	27	1	3.18	26	90	46	5	14.00	27	1	8.25	26	25
1300	0	2.46	20	2	4.82	12	80	41	4	9.22	20	3	5	9	22.2
1600	0	5.48	29	4	8.4	17	70	33	7	14.30	29	4	8.47	17	17.6
1900	0	3.42	40	4	15.1	44	96	44	11	18.66	40	10	22.5	44	66.7
2200	0	1.06	20	5	16.22	36	100	49	20	20	20	26	26	26	100
2500	0	1.9	31	7	21.42	40	100	47	17	23	31	30	33	37	100
2800	0	0.28	1	8	27.96	54	100	50							
3100	0	0.4	1	11	35.32	71	100	50							

Table 1. Experimentation report.

their computational time. All experiments have been processed on a 1.60 GHz Intel Xeon processor with 2 Gbytes of RAM. For each problem size, 50 problem instances have been considered.

To generate data for the single machine scheduling problem, we reuse the method introduced in [5]. The processing times of jobs correspond to uniform random variables between [1,50]. Each r_i is represented by an uniform random variable in the range $[0, \alpha \times \sum_{i=1}^n p_i]$, where α is a parameter allowing to adjust the distribution of r_i . Five values of α were considered: $\{0.2; 0.4; 0.6; 0.8; 1\}$. Similarly, each d_i is represented by an uniform random variable in the range $[(1-\beta) \times a \times \sum_{i=1}^n p_i, a \times \sum_{i=1}^n p_i]$. The parameter $a \in \{100\%, 110\%\}$ allows to adjust the maximal temporal margin associated to jobs and the parameter β controls the dispersion of d_i . Five values of β were considered $\beta \in \{0.2; 0.4; 0.6; 0.8; 1\}$. To ensure the coherence among the release date, the due date and the processing time of each job, any d_i smaller than $r_i + p_i$ has been updated to this value. For each problem size, combining each possible value of α , β and a, we obtain 50 problem instances.

Using those generating principles, we provide instances for single machine problems having $100,\,400,\,700,\,1000,\,1300,\,1600,\,1900,\,2200,\,2500,\,2800$ and 3100 jobs. For each problem size, Table 1 reports the following indicators :

- the min/mean/max cpu time for the Carlier's procedure (T_{Car}) ;
- the min/mean/max cpu time for running ILP3 (T_{ILP}) ;
- the percentage of times $T_{ILP} > T_{Car}$ (%W);
- the number of Schrage-optimal instances (#S);
- the min/mean/max cpu time for the Carlier's procedure only considering the hard instances $(T_{\text{Car}}^{\text{hard}})$;

- the min/mean/max cpu time for running ILP3 only considering the hard instances (T_{ILP}^{hard}) ;
- the percentage of times $T_{ILP}^{\text{hard}} > T_{\text{Car}}^{\text{hard}} \ (\%W^{\text{hard}}).$

The branch-and-bound procedure of Carlier [2] intensively uses the Schrage's algorithm in order to build up, at each node of the search tree, a job sequence. The local optimality of this job sequence is easily checked using a sufficient condition of optimality. We remark that it sometimes happens that the first Schrage's job sequence (i.e., the one computed at the root node considering the initial instance data) is optimal. In this case, we say that the problem instance is Schrage-optimal and we know that the Carlier's procedure only develops a single node for returning the optimal L_{max} value. As displayed in Table 1, this situation occurs many time in our experiments since, when $n \geq 2200$, we observe that the generating procedure often fails to provide instances which are not Schrage-optimal (i.e., #S is close to 50).

This is why we decided to provide Cplex with the initial Schrage's sequence, intending to decrease the computational effort needed to solve ILP3. Nevertheless, we highlight that in the case where the problem instance is Schrage-optimal, Cplex has still to prove the optimality of the sequence using LP techniques (while the Carlier's procedure is able to immediately deduce the optimality). This observation explains the first part of Table 1. Indeed, $T_{\rm Car}$ progressively increases as n increases until n=1900, which is the limit where #S becomes very close to 50, and then $T_{\rm Car}$ decreases when n increases. Regarding now T_{ILP} , we observe that the mean CPU time always increases as n increases, even when $\#S\cong 50$, since proving the optimality of the Schrage's sequence becomes more and more time expensive. Beyond this observation, we also observe that T_{ILP} is good for small instances (even better in average than $T_{\rm Car}$ when $n\le 700$) and remains acceptable for largest ones.

In order to refine the analysis, we decided to discard the Schrage-optimal problem instances from our statistics, only focusing on the hardest ones (see second part of Tab. 1). This time, we observe that the ILP approach is less time expensive than the Carlier's procedure in most of the cases, at least until n=1600. This seems to indicate that the ILP approach becomes competitive only when the problem instances are hard (*i.e.*, the number of nodes that the Carliers's procedure develops is large), although for large n values, it is difficult to make an opinion due to the low number of hard instances that were generated.

Another limit of the ILP approach concerns the size of the problem instances that it is able to cope with. Indeed, even if Cplex succeeds in solving instances having more than 4000 jobs, we observe that when there are many top jobs in the instance, the number of binary variables increase drastically and Cplex may run out of memory. On our test platform, we observe that this situation often occurs for instances of 4500 jobs having more than 300 top jobs (in this case, the corresponding reduced MIP has around 9000 rows and 700 000 columns). Conversely, the Carlier's procedure is not affected by the number of non-top jobs and is able to solve instances with more than 10 000 jobs without any problem.

7. Conclusion

This paper proposes an efficient ILP formulation for solving the general $1|r_j|L_{\rm max}$ problem. The experiments tend to demonstrate that our ILP approach is well suited to tackle medium-size instances, particularly when they are not Schrage-optimal (i.e., hard to solve). We also point out that, to the best of our knowledge, there does not exist any other ILP approach that can be compared with the clever branch-and-bound procedure of Carlier in term of efficiency, although we recognize that the Carlier's procedure remains the best-suited technique to solve efficiently $1|r_j|L_{\rm max}$ in most of the situations. Beyond these observations, we think that the interest of our ILP approach lies in its ability to be easily adapted for considering different objective function for the SMSP (e.g., the minimization of the number of late jobs) or, hopefully, for tackling more complex scheduling environment (e.g., parallel machines or job shop) with similar objective.

References

- [1] C. Briand, H.T. La and J. Erschler, Une approche pour l'ordonnancement robuste de tâches sur une machine, 4ème Conférence Francophone de MOdélisation et SIMulation (MOSIM'03). Toulouse, France (2003) 205–211.
- [2] J. Carlier, The one-machine sequencing problem. Eur. J. Oper. Res. 11 (1982) 42–47.
- [3] J. Erschler, F. Roubellat and J.-P Vernhes, Characterizing the set of feasible sequences for n jobs to be carried out on a single machine. Eur. J. Oper. Res. 4 (1980) 189–194.
- [4] 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.
- [5] A.M. Hariri and C.N. Potts, An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Appl. Math.* **5** (1983) 99–109.
- [6] J.K. Lenstra, A.H.G. Rinnooy Han and P. Brucker, Complexity of machine scheduling problems. Ann. Discrete Math. 1 (1977) 343–362.