

# A robust approach for the single machine scheduling problem

Cyril Briand\*, H. Trung La, Jacques Erschler

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

---

## Abstract

This paper describes a robust approach for the single machine scheduling problem  $1|r_i|L_{\max}$ . The method is said robust since it characterizes a large set of optimal solutions allowing to switch from one solution to another, without any performance loss, in order to face the potential disruptions which occur during the schedule execution. It is based on a dominance theorem that characterizes a set of dominant sequences, using the interval structure defined by the relative order of the release and the due dates of the jobs. The performance of a set of dominant sequences can be determined in polynomial time by computing the most favorable and the most unfavorable sequences associated with each job, with regard to the lateness criterion. A branch and bound procedure is proposed which modifies the interval structure of the problem in order to tighten the dominant set of sequences so that only the optimal sequences are conserved.

*Key words:* Scheduling, robustness, sequential flexibility, interval structures, pyramids.

---

## 1 Introduction

Robust scheduling methods aim to face uncertainties that usually arise during schedule execution. The goal is to achieve either solution robustness or quality robustness [28] meaning that a solution or its quality should be stable for a wide range of possible execution scenarios. The execution scenarios are usually assumed to be captured using a scenario model which associates either a probability distribution or a fuzzy set or an interval or a set of possible

---

\* Corresponding author.

*Email addresses:* [briand@laas.fr](mailto:briand@laas.fr) (Cyril Briand), [htla@laas.fr](mailto:htla@laas.fr) (H. Trung La), [erschler@laas.fr](mailto:erschler@laas.fr) (Jacques Erschler).

values [26,8] to the schedule parameters (release dates, due dates, processing times, ...). Therefore, robust scheduling methods can be classified according to the kind of model they use, the kind of robustness (solution or quality) they aim to achieve and the way they operate.

Following these classification criteria, several state-of-the-art papers [36,18,28,29] have pointed out the main classes of robust scheduling. As we only sum up here their main features, the reader should refer to these studies for more detailed descriptions.

*Reactive scheduling* is a kind of robust scheduling approach that consider disruptions only during the on line scheduling phase: a decision is made at each event occurrence, depending on the environment context. During the decision-making process, some priority rules are used which are assumed to be quite efficient with regard to a given performance objective. Commonly, a reactive approach uses neither a baseline schedule nor a scenario model. Therefore, while that kind of approach allows to face a wide range of disruptions (machine breakdown, processing time, release or due date fluctuations, ...), it does not ensure a good performance and does not provide decision-makers with a long term view of the schedule.

This is why *predictive-reactive scheduling* methods are often preferred. Indeed, those methods propose to use a baseline schedule computed during the off line scheduling phase. That schedule is repaired at appropriate moments of the on line phase in order to react against disruptions. Predictive-reactive approaches can be distinguished according to the repair moment (continuously, periodically, ...) and the way the initial baseline schedule is repaired (local schedule adaptation [41], total rescheduling [40], partial rescheduling [38], match-up rescheduling [39,2], ...). Predictive-reactive approaches usually yield better schedules than pure reactive ones since having a baseline schedule allows to improve the quality of the real time schedule decisions from the performance point of view. Nevertheless, because they do not take any scenario model into account, the quality and the stability of the initial baseline schedule can not be ensured a priori.

As pointed out by few authors [44,28], having a stable forecast of a project schedule allows a better human organization since decisions can be advantageously anticipated before the schedule implementation. This assertion has motivated the design of *proactive-reactive scheduling* methods which aim at anticipating, during the off line scheduling, the potential disruptions arising from the schedule environment. A first class of proactive-reactive approaches tends to obtain the solution robustness by computing a unique feasible solution having rather good performance for any execution scenario compatible with the scenario model [32,17,31]. On the other side, some proactive-reactive approaches tend to prioritize the quality robustness: a set of schedule solutions

is computed so that the performance can be ensured whatever the selected solution is. In this second class, a temporal flexibility [15,25,19,34,36,30,42] or a sequential flexibility [45,37,7,24,4] is inserted into an initial deterministic solution in order to protect it against unforeseen events. Indeed, a solution which is sequentially or temporally flexible characterizes a set of solutions that can be used in the on line phase. By switching from one solution to another when disruptions occur, the scheduler can control the performance degradation.

Several authors [10,22,26] have pointed out that it exists a trade-off between performance and robustness. An optimal schedule is usually robust with regard to a small set of disruptions. If one wants to protect the schedule against a larger set of disruptions then a degradation of its quality is necessary. The same assertion can also be found in robust optimization theory [6].

This paper focuses on proactive-reactive scheduling methods where sequential flexibility is used as a way to protect a solution against uncertainties. To handle sequential flexibility, a partial order is classically used. As defined in [14], a partial order  $P$  is defined by a pair  $P = (X, \preceq_P)$  where the binary relation  $\preceq_P$  on  $X \times X$  is reflexive, antisymmetric and transitive.

The notion of *group sequence* [43] gives an interesting partial order. It allows the characterization of a set of solutions specifying, for each resource, a sequence of groups of tasks. The execution order of the tasks within a group is free. The main advantage of such a partial order lies on the capability to perform a worst case analysis, without solution enumeration, in order to determine the quality of the set of solutions with regard to a regular criterion. Another advantage is that the number of characterized sequences can be easily computed, without solution enumeration, in order to quantify the flexibility. Of course, the larger the set of solutions, the worse its quality. The notion of group sequence has been widely used in the field of shop scheduling, intending to determine a family of solutions to be used during the schedule real time execution [35,4,9,5,45,7,33].

While the concept of group of tasks allows to exhibit a sequential flexibility, this flexibility is rather restricted. Indeed, that concept does not allow to characterize solutions when they correspond to permutations of tasks which are not adjacent on the same resource. For instance, the solutions  $i \prec S \prec j$  and  $j \prec S \prec i$ , where  $S$  is a set of tasks and  $i$  and  $j$  are two tasks, can not be represented together within a same group sequence.

Furthermore, as group sequences are classically built up without considering any uncertainty model, they are not robust a priori. Thus, computing a group sequence is certainly helpful in order to react to unforeseen events occurring in the on line phase, but it seems difficult to know in advance the set of disruptions that the set of solutions is insensitive to.

In this paper, the focus is put on the characterization of a large set of efficient schedules for single machine scheduling problems  $1|r_i|L_{\max}$ , while lessening the previous drawbacks. We highlight that, although single machine scheduling problems are somewhat academic, methods that allow to solve them have been often useful to tackle more realistic environments, like job shop problems [1,13,16]. We believe that the same assertion can be made for the robust framework presented below.

The paper is organized as follows. First some basic notions related to the analysis of interval structures are recalled since they are used for the definition of the partial order. Then a particular dominance theorem, stated in the early eighties, is presented. That theorem allows to characterize a set of dominant sequences and we show how both the number of characterized sequences and the worst performance of the dominant set can be efficiently computed, while avoiding the complete enumeration of the sequences. We also study the sensitivity of the dominant set of sequences relatively to a set of execution scenarios. Finally, a branch and bound procedure is detailed which aims at characterizing a large subset of optimal sequences by pruning the set of dominant sequences.

## 2 Interval structures and basic concepts

An interval structure is defined by a couple  $\langle I, C \rangle$  with  $I = \{i_1, \dots, i_n\}$  a set of intervals and  $C$  a set of constraints over  $I \times I$ . Each interval  $i_j$  is defined by its lower and upper bounds  $x_j$  and  $y_j$ . Any constraint between two intervals  $i_j$  and  $i_k$  can be expressed either by specifying a total order relation among the lower and upper bounds of the intervals or by directly using the relations of the algebra proposed by Allen [3] (see Figure 1).

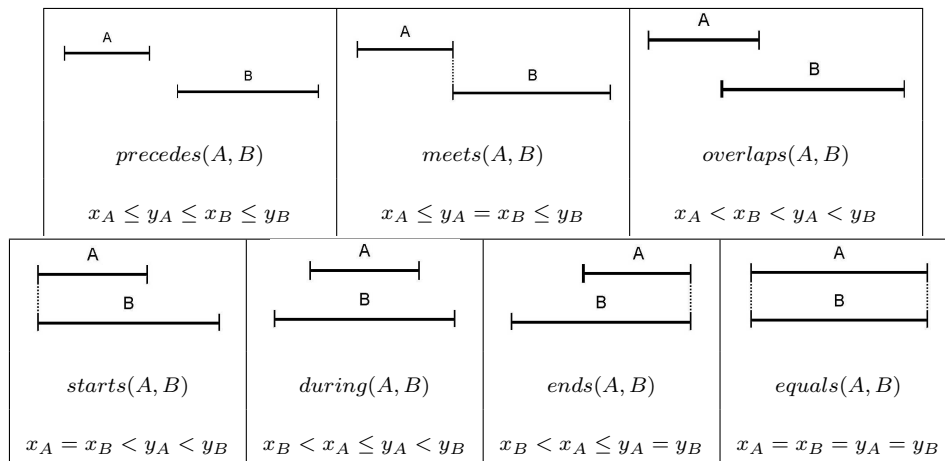


Fig. 1. Allen's relations

For instance, let us consider the interval structure with  $I = \{A, B, C, D, E, F, G\}$  given in Figure 2. We assume that the set of constraints  $C$  is defined by a total order such that  $(x_A = x_B = x_G) < (x_C = x_D) < y_C < (y_B = x_F) < x_E < y_A < (y_D = y_E) < y_F = y_G$ . The set of equivalent Allen's relations is represented in the Table given in Figure 3. Such a table can be computed with a time complexity  $O(n(n-1)/2)$ ,  $n$  being the number of intervals.

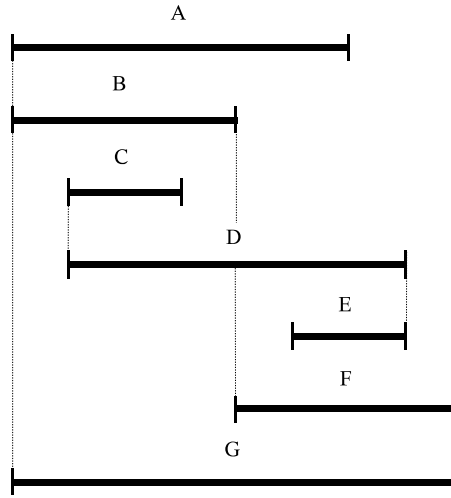


Fig. 2. An interval structure example

	A	B	C	D	E	F	G
A	<i>equals(A, A)</i>	<i>starts(B, A)</i>	<i>during(C, A)</i>	<i>overlaps(A, D)</i>	<i>overlaps(A, E)</i>	<i>overlaps(A, F)</i>	<i>starts(A, G)</i>
B	<i>starts(B, A)</i>	<i>equals(B, B)</i>	<i>during(C, B)</i>	<i>overlaps(B, D)</i>	<i>precedes(B, E)</i>	<i>meets(B, F)</i>	<i>starts(B, G)</i>
C	<i>during(C, A)</i>	<i>during(C, B)</i>	<i>equals(C, C)</i>	<i>starts(C, D)</i>	<i>precedes(C, E)</i>	<i>precedes(C, F)</i>	<i>during(C, G)</i>
D	<i>overlaps(A, D)</i>	<i>overlaps(B, D)</i>	<i>starts(C, D)</i>	<i>equals(D, D)</i>	<i>ends(E, D)</i>	<i>overlaps(D, F)</i>	<i>during(D, G)</i>
E	<i>overlaps(A, E)</i>	<i>precedes(B, E)</i>	<i>precedes(C, E)</i>	<i>ends(E, D)</i>	<i>equals(E, E)</i>	<i>during(E, F)</i>	<i>during(E, G)</i>
F	<i>overlaps(A, F)</i>	<i>meets(B, F)</i>	<i>precedes(C, F)</i>	<i>overlaps(D, F)</i>	<i>during(E, F)</i>	<i>equals(F, F)</i>	<i>ends(F, G)</i>
G	<i>starts(A, G)</i>	<i>starts(B, G)</i>	<i>during(C, G)</i>	<i>during(D, G)</i>	<i>during(E, G)</i>	<i>ends(F, G)</i>	<i>equals(G, G)</i>

Fig. 3. Allen's relations for the intervals  $\{A, B, C, D, E, F, G\}$

*Top* and *base* [21] are two interesting notions related to the concept of interval structure.

**Definition 1.** A *top* of an interval structure  $\langle I, C \rangle$  is an interval  $t \in I$  such that  $\forall i \in I$  the Allen's relation *during*( $i, t$ ) never holds.

**Definition 2.** A *base* of an interval structure  $\langle I, C \rangle$  is an interval  $b \in I$  such that  $\forall i \in I$  the Allen's relation *during*( $b, i$ ) never holds.

These notions of top and base can be respectively used to define the concepts of *t-pyramid* and *b-pyramid*.

**Definition 3.** Given a top  $t_\alpha$ , a *t-pyramid*  $P_\alpha$  related to  $t_\alpha$  is the set of intervals  $i \in I$  such that *during*( $t_\alpha, i$ ).

**Definition 4.** Given a base  $b_\alpha$ , a *b-pyramid*  $P_\alpha$  related to  $b_\alpha$  is the set of intervals such that *during*( $i, b_\alpha$ ).

For illustration, let us consider the interval structure given in Figure 2. It has

three tops  $\{C, D, E\}$  and four bases  $\{A, B, F, G\}$ . The involved t-pyramids are  $P_C = \{B, A, G\}$ ,  $P_D = \{G\}$  and  $P_E = \{F, G\}$ , and the b-pyramids are  $P_A = \{C\}$ ,  $P_B = \{C\}$ ,  $P_F = \{E\}$  and  $P_G = \{C, D, E\}$ .

The concept of b-pyramid has been already used in the scheduling literature in order to determine a sufficient condition of optimality for the  $F2|prmu|C_{\max}$  problem [11]. That condition allows to characterize a large subset of optimal sequences which necessarily includes any Johnson's sequences together with numerous other optimal job sequences. In the following, we focus on the t-pyramid concept which gives a dominant order for the  $1|r_j|L_{\max}$  problem.

### 3 A dominance theorem for the single machine problem

The following dominance theorem [20] has been stated in the main aim to prune the solution space associated with deterministic single machine scheduling problems, so that finding an optimal solution becomes less time expensive. Later on, we show how that theorem can also be used in a robust scheduling point of view.

A single machine problem  $V$  consists of a set  $T$  of  $n$  jobs to be scheduled on a single disjunctive resource. A start time  $s_j$  has to be found for each job  $j$ . The release date  $r_j$  ( $s_j \geq r_j$ ), the due date  $d_j$  and the processing time  $p_j$  of each job are known. The studied dominance is relative either to the admissibility of the problem (i.e.  $\forall j \in T, s_j + p_j \leq d_j$ ) or to the minimization of the maximum lateness  $L_{\max} = \max(s_j + p_j - d_j)$  or to the minimization of the maximum tardiness  $T_{\max} = \max(0, s_j + p_j - d_j)$ .

The hypothesis frame used by the authors to study the dominance takes into account only the relative order of the release dates  $r_j$  and due dates  $d_j$  of the jobs. Therefore the processing time  $p_j$  as well as the explicit values of  $r_j$  and  $d_j$  are not used. In other words, whatever the values of  $r_j$ ,  $d_j$  and  $p_j$  are, the following results are valid as long as the relative order of the release and due dates is kept unchanged.

An interval structure  $\langle I_V, C_V \rangle$ , associated with a problem  $V$ , contains interval  $i_j = [r_j, d_j] \in I_V$  for each job  $j$ . To characterize a dominant set of sequences, the authors use the notions of tops and t-pyramids related to  $\langle I_V, C_V \rangle$ .

It is assumed that the tops are indexed according to the ascending order of their release dates or, in case of equality, according to the ascending order of their due dates. When both their release dates and due dates are equal, the tops are indexed in an arbitrary order. Thus, if  $t_\alpha$  and  $t_\beta$  are two tops then

$\alpha < \beta$  if and only if  $(r_{t_\alpha} \leq r_{t_\beta}) \wedge (d_{t_\alpha} \leq d_{t_\beta})$ . The t-pyramid  $P_\alpha$  corresponds to the pyramid that the top  $t_\alpha$  characterizes. The functions  $u(j)$  (resp.  $v(j)$ ) indicates the index of the first (resp. the last) t-pyramid to which the job interval  $i_j$  belongs.

The theorem can now be stated.

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

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

Theorem 1 defines a partial order since on the one hand, it imposes some precedence relations among the tops of the interval structure (i.e.  $t_k \prec t_{k+1}$ ) and on the other hand, between the non-top jobs and the tops (i.e.  $t_{u(j)-1} \prec j \prec t_{v(j)+1}$ ). However, since the jobs sequenced before (resp. after) a top have to be ordered in the ascending order of their release dates  $r_j$  (resp. in the ascending order of their due dates  $d_j$ ), the partial order is restricted.

In order to illustrate the theorem, let us focus on a class of problem having seven jobs so that the relative order among the release dates  $r_j$  and due dates  $d_j$  of the jobs is  $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 < (r_5 = r_7) < d_6 < d_5 < d_1 < d_7$ . The interval structure associated with that example is shown on Figure 4, as well as the tops (in bold) and the t-pyramids.

There are four tops:  $t_1 = 2$ ,  $t_2 = 4$ ,  $t_3 = 5$  and  $t_4 = 7$ , which characterize four t-pyramids:  $P_1 = \{1, 3, 6\}$ ,  $P_2 = \{1, 6\}$ ,  $P_3 = \{1\}$  and  $P_4 = \emptyset$  (in accordance with the definition a top does not belong to the pyramid it characterizes).

Whatever the real values of  $r_i$  and  $d_i$  are (provided they are compatible with

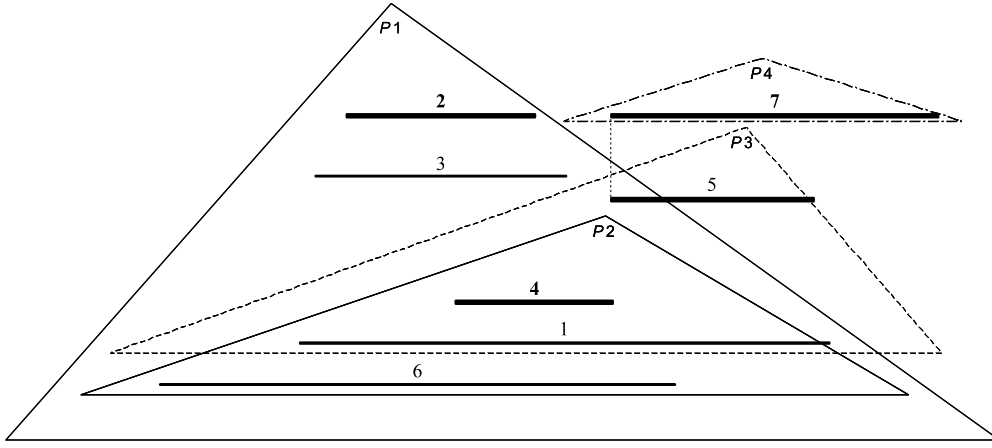


Fig. 4. The interval structure of the problem

the previous interval structure), theorem 1 characterizes twenty four dominant sequences (over  $7! = 5040$ ):

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

#### 4 A flexibility measure

An interesting property of theorem 1 is that the number of dominant sequences, included in the set  $S_{\text{dom}}$  which is characterized by the theorem, can be computed according the following formula [20], without the requirement of any sequence enumeration.

$$\text{card}(S_{\text{dom}}) = \prod_{q=1}^N (q+1)^{n_q}$$

where  $n_q$  is the number of non-top jobs belonging to exactly  $q$  pyramids and  $N$  is the total number of pyramids.

In order to illustrate this formula, we consider the previous example. There are three non-top jobs  $\{1, 3, 6\}$ . Job 3 belongs to a single t-pyramid, job 6 belongs to exactly two t-pyramids and job 1 belongs to exactly three t-pyramids. The application of the formula gives  $\text{card}(S_{\text{dom}}) = (1+1)^1 \cdot (2+1)^1 \cdot (3+1)^1 = 24$ .

## 5 A performance measure

Theorem 1 associates to each interval structure corresponding to a class of problem  $V$ , a set of dominant sequences  $S_{\text{dom}}$ . This section shows how the performance of that set can be measured regarding the maximum lateness criterion.

In the following, the focus is put on the determination of the values  $L_j^{\max}$  and  $L_j^{\min}$ , corresponding respectively to the worst and the best lateness of a job  $j$  among all the sequences of  $S_{\text{dom}}$ . As shown here under, the computation of those values can be performed in time complexity  $O(n \log n)$  by determining the most unfavorable sequence  $Seq_j^{\max}$  and the most favorable sequence  $Seq_j^{\min}$ , in  $S_{\text{dom}}$ , for each job  $j$ .

Let us underline that the determination of  $Seq_j^{\max}$  and  $Seq_j^{\min}$  only requires the knowledge of the relative order of the release and due dates of the jobs, although the computation of  $L_j^{\max}$  and  $L_j^{\min}$  takes the explicit values of  $r_j$ ,  $p_j$  and  $d_j$  into account. We also recall that in the following,  $u(j)$  and  $v(j)$  respectively indicates the index of the first and the last t-pyramid to which the job  $j$  belongs.

### 5.1 Determination of $Seq_j^{\min}$

The most favorable sequence  $Seq_j^{\min} \in S_{\text{dom}}$  for  $j$  is the sequence in which  $j$  is completed as early as possible. It is determined using the Jackson's rule.

We denote  $Pred_j^{\min}$  the set of jobs such that  $v(k) < u(j)$ , i.e. the jobs which necessarily precedes  $j$  in any dominant sequence. Then the following theorem is proved.

**Theorem 2.** *The most favorable sequence  $Seq_j^{\min} \in S_{\text{dom}}$  for a job  $j$  is  $\sigma_1 \prec t_1 \prec \dots \prec \sigma_{u(j)-1} \prec t_{u(j)-1} \prec j$  so that the jobs of  $\sigma_k = \{i \in Pred_j^{\min} | u(i) = k\}$  are sequenced in the ascending order of their release dates (see Figure 5).*

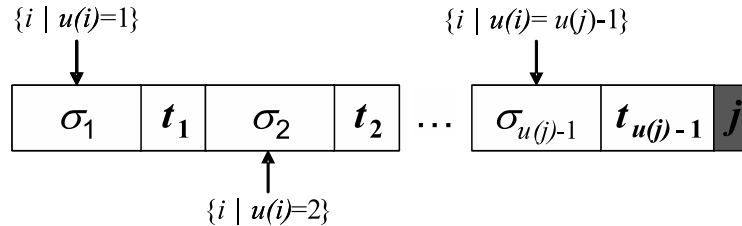


Fig. 5. Structure of  $Seq_j^{\min}$

*Proof.* First, it is obvious that only the jobs in  $Pred_j^{\min}$ , which necessarily

precedes  $j$  in any dominant sequence, have to be considered since sequencing a job, not in  $Pred_j^{\min}$ , before  $j$  can only increase the lateness of  $j$ . So finding  $L_j^{\min}$  amounts to the minimization of the total completion time of the jobs of  $Pred_j^{\min}$ . Since the lateness of the jobs in  $Pred_j^{\min}$  do not matter, we can use the Jackson's rule which states that the sequence ordering the jobs according to the ascending order of their release dates is optimal for  $1|r_i|C_{\max}$ . Indeed, the Jackson's rule respects theorem 1 since it assigns each job  $i$  to its pyramid  $P_{u(i)}$ , just before top  $t_{u(i)}$ .  $\square$

Let us return to the example given in Figure 4. We focus on job 5. This job is a top which characterizes pyramid  $P_3$ . The set  $Pred_5^{\min}$  is made of the jobs  $k$  such that  $v(k) < 3$ , i.e.  $Pred_5^{\min} = \{2, 3, 6, 4\}$ . According to the theorem, the most favorable sequence for job 5 is  $Seq_5^{\min} = 6 \prec 3 \prec 2 \prec 4 \prec 5$ .

## 5.2 Determination of $Seq_j^{\max}$

In order to compute the worst possible value of  $L_j^{\max}$  of a job  $j$ , we determine the most unfavorable sequence  $Seq_j^{\max} \in S_{\text{dom}}$  for  $j$ , i.e. the sequence in which  $j$  is completed as late as possible. That can be done also with a temporal complexity  $O(n \log n)$ .

We denote  $Pred_j^{\max}$  the set of jobs  $k$  such that  $u(k) \leq v(j)$  (the jobs such that  $u(k) > v(j)$  are necessarily sequenced after  $j$  in any dominant sequence). Before to give the general way to build up  $Seq_j^{\max}$ , we need to first focus our attention on the case where the problem has only one t-pyramid.

**Theorem 3.** *Given an interval structure associated to a problem  $V$ , having a single t-pyramid  $P$  of top  $t$ , the most unfavorable sequence  $Seq_j^{\max} \in S_{\text{dom}}$  with regard to the lateness of job  $j \in P$  is  $A \prec t \prec B \prec j$  where the jobs of  $A = \{i \in P | d_i > d_j\}$  are sequenced in the ascending order of their release dates and the jobs of  $B = \{i \in P | d_i \leq d_j\}$  are sequenced in the ascending order of their due dates.*

*Proof.* The problem of maximizing the lateness of  $j$  amounts to the maximization of the completion time of  $j$  since the lateness of the other jobs do not matter. Therefore, we take interest in sequences having the form  $S = A \prec t \prec B \prec j$ , the jobs of  $A$  (resp. of  $B$ ) being ordered according to the ascending order of their release dates (resp. according to the ascending order of their due dates). The completion time of  $j$  in  $S$  is  $C_j^S = \max(C_A, r_t) + p_t + \sum_{k \in B} p_k + p_j$ . We assume below that the conditions of the theorem 3 are satisfied, i.e.  $A = \{i \in P | d_i > d_j\}$  and  $B = \{i \in P | d_i \leq d_j\}$ .

Then it is easy to verify that if a job  $i$  in  $A$ , is moved after the top  $t$  then the

completion time of  $j$  can never increase. Indeed, in this case, because  $d_i > d_j$  the new sequence  $S'$  has the form  $A' \prec s \prec B \prec j \prec i$  (with  $A' = A - \{i\}$ ) and obviously  $C_j^{S'} \leq C_j^S$ .

Similarly, one can verify that if a job  $i \in B$  is moved before the top  $t$  then the completion time of  $j$  can not increase. Indeed, according to theorem 1, the new sequence  $S'$  is in the form  $A_1 \prec i \prec A_2 \prec t \prec B' \prec j$  (with  $A = A_1 \prec A_2$  and  $B' = B - \{i\}$ ) and one can checked that  $C_j^S - C_j^{S'} = \max(C_A, r_t) - \max(C_{A_1 \prec i \prec A_2}, r_t) + p_i$  is greater or equal than zero because  $C_A - C_{A_1 \prec i \prec A_2} \leq p_i$ . Hence the theorem.  $\square$

We can now consider the general case where the interval structure attached to the problem has several t-pyramids.

**Theorem 4.** *The most unfavorable sequence  $Seq_j^{\max} \in S_{\text{dom}}$  for a job  $j$  is  $t_1 \prec \sigma_1 \prec \dots \prec t_{v(j)-1} \prec \sigma_{v(j)-1} \prec A \prec t_{v(j)} \prec B \prec j$  so that the jobs of  $\sigma_k = \{i \in Pred_j^{\max} | v(i) < v(j) \text{ and } v(i) = k\}$  are sequenced in the ascending order of their due dates, the jobs of  $A = \{i \in Pred_j^{\max} | v(i) \geq v(j) \text{ and } d_i > d_j\}$  are sequenced in the ascending order of their release dates, and the jobs of  $B = \{i \in Pred_j^{\max} | v(i) \geq v(j) \text{ and } d_i \leq d_j\}$  are sequenced in the ascending order of their due dates (see Figure 6).*

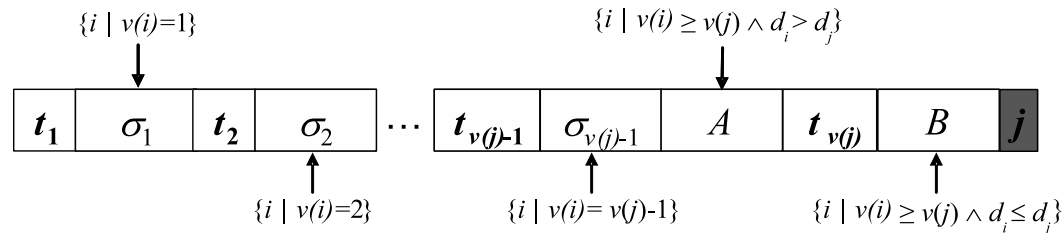


Fig. 6. Structure of  $Seq_j^{\max}$

*Proof.* The problem of maximizing the lateness of  $j$  amounts to the maximization of the completion time of  $j$  since the lateness of the other jobs do not matter. Obviously, the greater the number of jobs sequenced before  $j$ , the greater becomes the completion time of  $j$ . So, according to theorem 1, only the jobs  $i \in Pred_j^{\max}$  such that  $u(k) \leq v(j)$  have to be considered. Furthermore, the later the jobs in  $Pred_j^{\max}$  are sequenced, the greater is the completion time of  $j$ . Therefore any job  $i \in Pred_j^{\max}$  has to be assigned, in accordance with theorem 1, either to  $P_{v(i)}$ , if  $v(i) < v(j)$ , or to  $P_{v(j)}$  otherwise. Thus the problem of maximizing the completion time of the jobs of  $Pred_j^{\max}$  amounts to  $v(j)$  independent problems of maximization, one for each t-pyramid. Maximizing the completion time of the jobs of  $P_\alpha$ , with  $\alpha \leq v(j) - 1$ , is easy because the worst completion time is obviously  $C_\alpha = r_{t_\alpha} + \sum_{i \in P_\alpha} p_i + p_{t_\alpha}$  which is obtained when the jobs are sequenced after the top (in the ascending order of their due

dates). Maximizing the completion time of the t-pyramid  $P_{v(j)}$  can be done according to theorem 3.  $\square$

In order to illustrate theorem 4, let us return to the example given in Figure 4. We focus our attention on job 6, and on its most unfavorable sequence  $S_6^{\max}$ . This job belongs to two pyramids characterized by tops 2 and 4, so  $v(6) = 2$ . The set  $Pred_6^{\max}$  is  $\{1, 2, 3, 4\}$ . From the previous theorem, one can deduce that  $Seq_6^{\max} = 2 \prec 3 \prec 1 \prec 4 \prec 6$ .

### 5.3 Lateness diagram

Given a problem  $V$  and its dominant set of sequences  $S_{\text{dom}}$ , it is interesting to lay out a visual representation that gives the best and the worst lateness for each job. For this purpose, the diagram of lateness is introduced which associates to each job  $i \in T$  an interval  $[L_i^{\min}, L_i^{\max}]$ . The bounds of this interval are respectively computed on the basis of  $Seq_j^{\min}$  and  $Seq_j^{\max}$ .

Figure 7 presents the lateness diagram associated to the problem of Table 1, stemmed from Carlier (1982). We underline that this problem matches the interval structure given in Figure 4. The most favorable and unfavorable sequences of each job are indicated above each bound of the intervals.

<i>Jobs</i>	$r_j$	$d_j$	$p_j$
1	10	44	5
2	13	25	6
3	11	27	7
4	20	30	4
5	30	43	3
6	0	34	6
7	30	51	2

Table 1  
A single machine scheduling problem

Given  $L_i^{\min}$  and  $L_i^{\max}$ , the optimal  $L_{\max}$  can be bounded as follows:

$$\max(L_j^{\min}) \leq L_{\max}^* \leq \max(L_j^{\max}), \forall j \in T$$

For instance, as depicted in Figure 7, one can deduce that  $-2 \leq L_{\max}^* \leq 11$ . These bounds have to be compared to the optimal lateness  $-1$ , which can

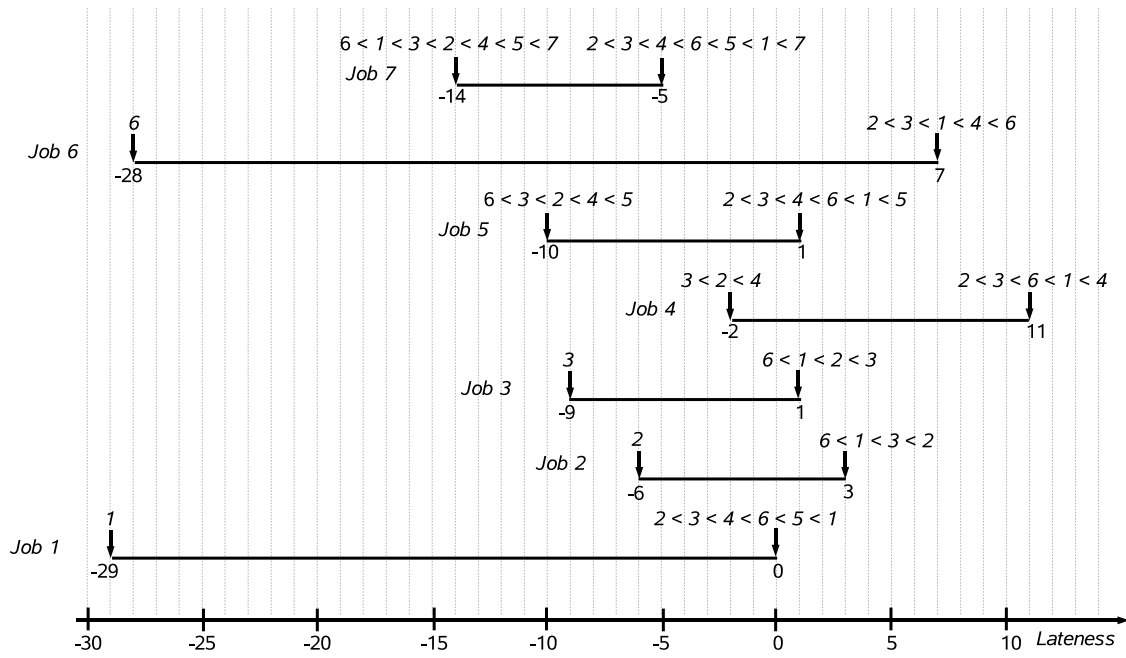


Fig. 7. Lateness diagram for the example of Table 1

be found using the branch and bound procedure of Carlier [12]. One can also remark that, for any sequence of  $S_{\text{dom}}$ , jobs 1 and 7 are never late.

## 6 An uncertainty model by intervals

As claimed in the introduction, the off-line characterization of a set of schedules allows to favor the robustness of the on line decision-making process since it is possible to face the disruptions by switching opportunely from a solution to another, while controlling the performance degradation. Nevertheless, given a set of schedule, one can also want to know in advance the range of disruptions the set of schedules is robust to. Indeed, that kind of feature allows decision-makers to size the sequential flexibility which is required in order to protect the schedule against a set of possible execution scenarios. We show below how, given an interval structure and an interval model of the uncertainties, it is possible to determine a set of schedules having a secured performance whatever the considered execution scenario is.

One interesting property of theorem 1 is that it is insensitive to variations of both the release and due dates, provided that their relative order is kept unchanged. Moreover, it is also insensitive to variations of processing times which are not considered inside the theorem. In section 5, it has been shown that the most favorable and unfavorable sequences  $Seq_j^{\min}$  and  $Seq_j^{\max}$ , regarding the lateness of a job, are also unchanged under the same assumptions. Then

the following property obviously stands.

**Property 1.** *Given a single machine scheduling problem  $V$ , having its set of potential execution scenarios characterized by an interval model in the form  $r_i \in [\underline{r}_i, \bar{r}_i]$ ,  $p_i \in [\underline{p}_i, \bar{p}_i]$  and  $d_i \in [\underline{d}_i, \bar{d}_i]$  then, if all the intervals  $[\underline{r}_i, \bar{r}_i]$  and  $[\underline{d}_i, \bar{d}_i]$  are disjointed, theorem 1 characterizes a set of dominant sequences  $S_{dom}$  for which a best and a worst lateness  $\underline{L}_j^{\min}$  and  $\bar{L}_j^{\max}$  can be ensured whatever the considered execution scenario is. Furthermore, whatever the considered execution scenario is, the set  $S_{dom}$  always involves at least one optimal solution.*

*Proof.* If all the intervals  $[\underline{r}_i, \bar{r}_i]$  and  $[\underline{d}_i, \bar{d}_i]$  are disjointed, then a total order exists among the release and due dates. Therefore theorem 1 can be applied in order to find a set of dominant sequences  $S_{dom}$  and one can ensure that it always involves at least one optimal solution, whatever the considered scenario is, provided it is coherent with the interval model. Given a job  $j \in V$ , section 5 has shown how to determine the most favorable and unfavorable sequences  $Seq_j^{\min}$  and  $Seq_j^{\max}$ . They are also independent of the real values of  $r_j$ ,  $p_j$  and  $d_j$ , since only the relative order among the  $r_j$  and  $d_j$  is used for their determinations. Given the sequence  $Seq_j^{\min}$  (resp.  $Seq_j^{\max}$ ) associated to a job  $j$ , a lower bound  $\underline{L}_j^{\min}$  (resp. an upper bound  $\bar{L}_j^{\max}$ ) of its lateness can be obviously determined by using the values  $\underline{r}_i$ ,  $\underline{p}_i$  and  $\bar{d}_i$  (resp.  $\bar{r}_i$ ,  $\bar{p}_i$  et  $\underline{d}_i$ ).  $\square$

This last property is of importance since it can be ensured that a set of dominant sequences has a performance which is a priori robust relatively to a set of possible execution scenarios. In other words, if the potential uncertainties are modeled by means of intervals associated with the release dates, the due dates and the processing times, then a best and a worst lateness can be computed for each job, provided that the interval  $[\underline{r}_i, \bar{r}_i]$  and  $[\underline{d}_i, \bar{d}_i]$  are disjointed.

The assumption that the intervals  $[\underline{r}_i, \bar{r}_i]$  and  $[\underline{d}_i, \bar{d}_i]$  are disjointed does not seem unrealistic. For instance, in a make-to-order context, it only implies on one hand, that the components which are required for the production are not delivered by the suppliers at the same time but in disjointed time windows, and on the other hand, that the order due dates are distributed in time according to disjointed time windows.

For illustration, let us consider the problem having the interval model described below (without loss of generality the processing times are assumed to be constant):

j	$[r_j, \bar{r}_j]$	$[\underline{d}_j, \bar{d}_j]$	$p_j$
1	[6,9]	[10,15]	4
2	[1,2]	[36,37]	5
3	[20,22]	[33,35]	8
4	[23,27]	[28,32]	6
5	[3,5]	[16,19]	7

This interval model characterizes 259200 different possible scenarios of execution. There are two tops (jobs 1 and 4) and theorem 1 determines twelve dominant sequences:

$2 \prec 5 \prec 1 \prec 3 \prec 4$ ,  $2 \prec 5 \prec 1 \prec 4 \prec 3$ ,  $5 \prec 1 \prec 2 \prec 3 \prec 4$ ,  $5 \prec 1 \prec 2 \prec 4 \prec 3$ ,  
 $5 \prec 1 \prec 3 \prec 4 \prec 2$ ,  $5 \prec 1 \prec 4 \prec 3 \prec 2$ ,  $2 \prec 1 \prec 5 \prec 3 \prec 4$ ,  $2 \prec 1 \prec 5 \prec 4 \prec 3$ ,  
 $1 \prec 5 \prec 2 \prec 3 \prec 4$ ,  $1 \prec 5 \prec 2 \prec 4 \prec 3$ ,  $1 \prec 5 \prec 3 \prec 4 \prec 2$ ,  $1 \prec 5 \prec 4 \prec 3 \prec 2$ .

The computation of the best and worst lateness for each job gives the lateness diagram of the Figure 8. The lateness values are robust with regard to any possible interval scenario which respects the initial interval model.

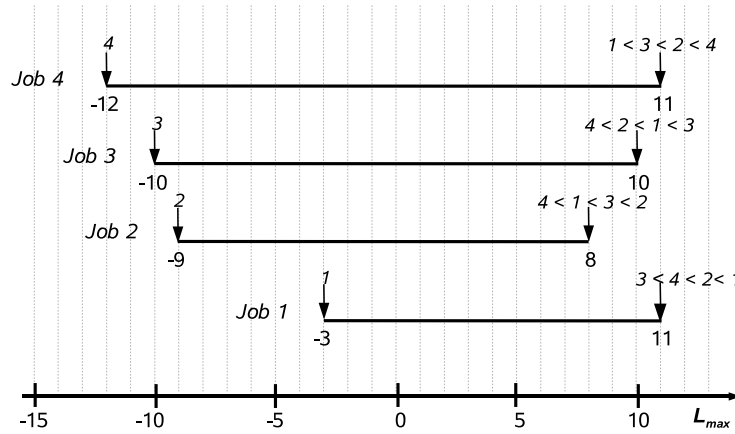


Fig. 8. Lateness diagram

## 7 A branch and bound procedure

### 7.1 General principles

Given an interval structure attached to a problem and its corresponding set of dominant sequences  $S_{dom}$ , it can be interesting to prune  $S_{dom}$  in order to delete bad sequences so that its worst performance (i.e  $\max(L_j^{max})$ ) can be

decreased. Indeed, such a feature is useful when searching for a good trade-off between flexibility and performance.

With this aim, given the initial interval structure  $\langle I_V, C_V \rangle$  of a problem  $V = \{j|r_j, d_j, p_j\}$ , one can explore the set of interval structures  $\langle I_{V'}, C_{V'} \rangle$  which are *compatible* with  $\langle I_V, C_V \rangle$ , i.e such that  $V' = \{j|r'_j \geq r_j, p'_j = p_j, d'_j \leq d_j\}$ .

For this purpose, we developed a branch and bound procedure. It is assumed that an upper bound of the worst lateness  $L$  is defined by decision-makers, with  $L \geq L_{\text{opt}}$  ( $L_{\text{opt}}$  being the optimal lateness of the considered problem  $V$ ). The goal of the procedure is to enumerate the interval structures  $\langle I_{V'}, C_{V'} \rangle$  (with  $V' = \{j|r'_j \geq r_j, p'_j = p_j, d'_j \leq d_j\}$ ) having a worst lateness less than or equal to  $L$ .

Each node of the exploration tree is evaluated by a lower bound and an upper bound of the lateness criterion. They respectively correspond to the values  $\max_{j \in T} L_j^{\min}$  and  $\max_{j \in T} L_j^{\max}$ , established as described in the section 5. The separation of a node  $i$  is stopped either when its lower bound is greater than  $L$  (i.e. all the characterized sequences have a maximal lateness greater than  $L$ ) or when its upper bound is less than or equal to  $L$  (i.e. all the characterized sequences have a maximal lateness less than or equal to  $L$ ). In the latter case, the node has to be memorized. A depth-first search strategy is adopted.

## 7.2 Separation and branching schemes

Given a node of the tree, the longest path  $C$ , associated to the sequence which gives to the worst lateness its value, is determined. On this path, two jobs are selected : the *pivot* job corresponding to a top, and a *free* non-top job which is sequenced either to the right or to the left of the pivot. In order to select the pivot and the free job, three cases are considered. Let  $j$  be the job such that  $L_j^{\max} = \max_{i \in T} L_i^{\max}$ .

- if  $j$  is a non-top job, then the pivot is the top of the pyramid having the index  $v(j)$ , the job being necessarily on  $C$ ; the free job is the immediate job being immediately located at the right of the pivot on  $C$ ;
- if  $j$  is a top, and if the t-pyramid it characterizes holds some non-top jobs, then the pivot is  $j$  and the free job is the job being immediately located at the left of the pivot on  $C$ ;
- if  $j$  is a top, and if the t-pyramid it characterizes does not hold any non-top jobs, then the pivot is the next top on  $C$  at the left of  $j$  having still some non-top jobs in its pyramid, the free job is the job immediately located at the right of the pivot on  $C$ .

Let respectively  $i$  and  $\alpha$  be the free job and the pivot. A binary separation scheme is considered:  $\alpha$  precedes  $i$  or  $i$  precedes  $\alpha$ . According theorem 1, those precedences can be achieved as follows:

- $\alpha \prec i$  is imposed by updating  $r_i$  so that  $r_i \leftarrow r_\alpha$  since  $\alpha$  being a top,  $r_j \geq r_\alpha$  induces that  $\alpha$  precedes  $j$  in any dominant sequence;
- similarly,  $i \prec \alpha$  is imposed by updating  $d_i$  so that  $d_i \leftarrow d_\alpha$ .

The root of the exploration tree is the initial interval structure of the problem. Each node  $N$  of the exploration tree has two children,  $N_{\alpha \prec i}$  and  $N_{i \prec \alpha}$  having their own interval structure and dominant set of sequences  $S_{\text{dom}}(N_{\alpha \prec i})$  and  $S_{\text{dom}}(N_{i \prec \alpha})$ , such that  $S_{\text{dom}}(N_{\alpha \prec i}) \cup S_{\text{dom}}(N_{i \prec \alpha}) = S_{\text{dom}}(N)$ . The child having the best upper bound is selected. The separation of a node stops either when it corresponds to a solution or when  $C$  only holds top jobs. In the latter case, the procedure backtracks to the last still unexplored node.

### 7.3 Example

In order to illustrate how the procedure works, let us return to the example of Table 1. Here we assume that the targeted lateness  $L$  is  $L_{\text{opt}} = -1$ . Our procedure passes by the following stages:

Stage 1: The node  $N_0$ , having its upper bound  $\max_{i \in T} L_i^{\text{max}} = L_4^{\text{max}} = 11 > L_{\text{opt}}$ , is separated. The sequence giving to  $L_4^{\text{max}}$  its value is  $2 \prec 3 \prec 6 \prec 1 \prec 4$ . The job 2 is selected as the pivot, and 3 as the free job. Thus two branchings are considered:  $r_3 \leftarrow r_2$  and  $d_3 \leftarrow d_2$  which give birth to the nodes  $N_1$  and  $N_2$  (see Figure 9).

The new lower and upper bounds of these nodes are evaluated as indicated in Figure 9. The node  $N_1$  having its lower bound equals to 0, hence greater than  $L_{\text{opt}}$ , is cut. Only  $N_2$  has to be developed.

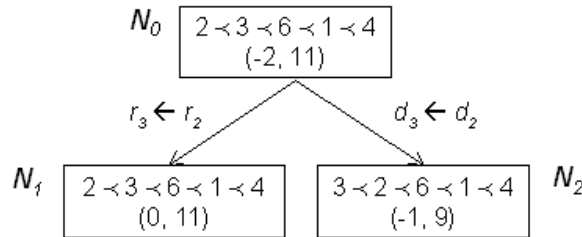


Fig. 9.  $N_0$

Stage 2: the longest path associated to  $N_2$  is  $3 \prec 2 \prec 6 \prec 1 \prec 4$ . Job 3 is selected as the pivot and job 6 as the free job. Two new branchings  $N_3$  and  $N_4$  are considered (see Figure 10): the one where  $r_6 \leftarrow r_3$  and the one where

$d_6 \leftarrow d_3$ . The next selected node is  $N_4$ .

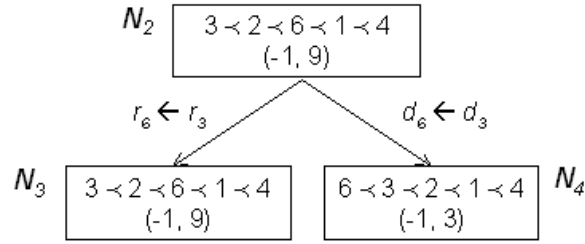


Fig. 10.  $N_2$

Stage 3: the longest path associated to  $N_4$  is  $6 \prec 3 \prec 2 \prec 1 \prec 4$ . Job 4 is selected as the pivot and job 1 as the free job. The two new branchings are  $N_5$  and  $N_6$ :  $r_1 \leftarrow r_4$  and  $d_1 \leftarrow d_4$ . The lower bound of  $N_6$  being greater than  $-1$ ,  $N_6$  is cut. Furthermore,  $N_5$ , having both its lower and upper bounds equal to  $-1$ , is an optimal node. Its interval structure characterizes two optimal sequences:  $6 \prec 3 \prec 2 \prec 4 \prec (1 - 5) \prec 7$ , where  $(1 - 5)$  is a group of permutable jobs. The node  $N_3$  has still to be developed since other optimal solutions can be found (see Figure 11).

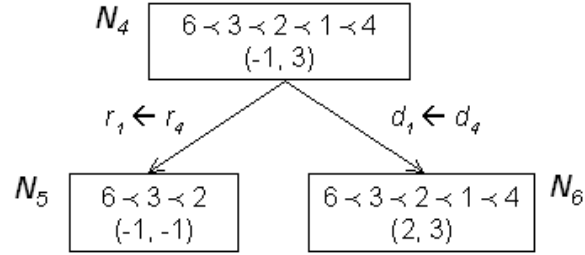


Fig. 11.  $N_4$

Stage 4: For  $N_3$ , the longest path is  $3 \prec 2 \prec 6 \prec 1 \prec 4$ . Jobs 2 and 6 are respectively chosen as pivot and free job. Two new nodes  $N_7$  and  $N_8$  have to be considered:  $r_6 \leftarrow r_2$  and  $d_6 \leftarrow d_2$ .  $N_8$  can be cut and the node  $N_7$  is developed in the next stage (see Figure 12).

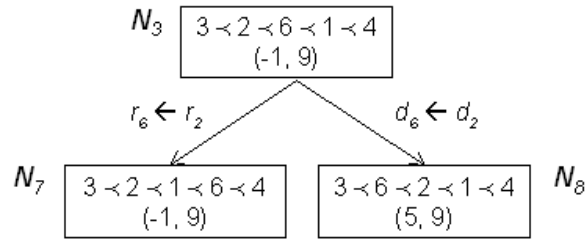


Fig. 12.  $N_3$

Stage 5: For  $N_7$ , the selected pivot is job 4 and the free job is 1. Two new nodes are created  $N_9$  et  $N_{10}$  corresponding to the updating :  $r_1 \leftarrow r_4$  and  $d_1 \leftarrow d_4$ .  $N_{10}$  can be cut and  $N_9$  is considered in the next stage (see Figure 13).

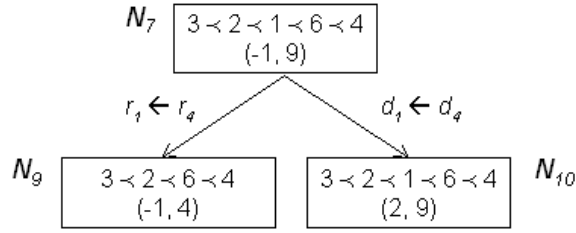


Fig. 13.  $N_7$

Stage 6: For  $N_9$ , the longest path is  $3 \prec 2 \prec 6 \prec 4$ . Jobs 6 and 4 are respectively chosen as the pivot and the free job. Two new nodes are obtained:  $N_{11}$  and  $N_{12}$  which can both be cut (see Figure 14). Since all the nodes have already been explored, the procedure stops. Only the node  $N_5$  is optimal.

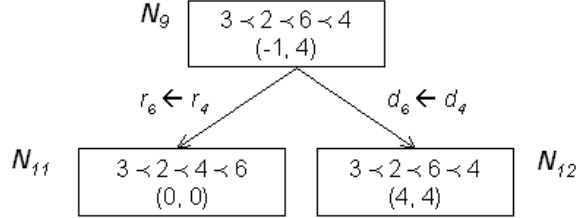


Fig. 14.  $N_9$

## 8 Computational experiments

In this section, we present the computational results of the procedure described above. We intend to show that it allows to characterize, in a short time, an impressive number of optimal solutions.

### 8.1 Random generator and experimentation strategy

To generate data for the single machine scheduling problem, we reuse the method introduced in [27]. The processing times of jobs correspond to uniform random variables between  $[1,100]$ . Each  $r_i$  is represented by an uniform random variable in the range  $[0, \alpha \times \sum_{i=1}^n p_i]$ , where  $\alpha$  is a parameter allowing to adjust the distribution of  $r_i$ . Four values of  $\alpha$  were considered:  $\{0.25; 0.5; 0.75; 1\}$ . 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  $\beta$  controls the dispersion of  $d_i$ . Four values of  $\beta$  were considered  $\beta \in \{0.25; 0.5; 0.75; 1\}$ . To ensure, for each job, the coherence between the release date, the due date and the processing time, any  $d_i$  smaller than  $r_i + p_i$  is updated to this value.

Using those generating principles, we generate instances for single machine problems of 10, 50, 100 and 500 jobs. In each case, we generated 320 instances, ten for each possible combination of the values of  $\alpha$ ,  $\beta$  and  $a$ .

For each instance, we take an interest in optimality searching by fixing the targeted lateness  $L$  to  $L_{\text{opt}}$ . The Carlier's branch and bound procedure has been used for computing  $L_{\text{opt}}$ .

Let us point out that, although the procedure is able to find all the admissible interval structures, the computational effort required to enumerate all of them is considerable for large problems. Nevertheless, as it is shown below, the number of dominant sequence that the first found interval structure characterizes is often very high. This is why, in each run, we decided to stop the branch and bound procedure as soon as a first admissible interval structure is found.

## 8.2 Summarized results

An overview of the results given by the procedure is presented in Table 2. The tests were executed on a 1.4 GHz Intel processor with 1 Gb RAM. We point out, for each problem class, the average CPU time consumed to find the first optimal interval structures, and the average number of optimal sequences that the first structures characterize. Let us notice that the CPU times given in the table *do not* include the time consumed by the algorithm of Carlier for computing  $L_{\text{opt}}$ .

		Tcpu (sec.)	Card( $S_{\text{dom}}$ )
10 jobs	Avg.	0.00018	7.27
	Max.	0.01	108
	Min.	0.001	1
50 jobs	Avg.	0.068	3.67E+22
	Max.	0.11	1.01E+25
	Min.	0.02	1
100 jobs	Avg.	0.43	1.87E+63
	Max.	0.631	5.51E+65
	Min.	0.22	36
500 jobs	Avg.	57.17	2.06E+303
	Max.	93.731	> 1.00 E.308
	Min.	25.138	3.1568E+12

Table 2  
Summarized results

Table 2 shows that the amount of time needed to solve problems with 10-50-100 jobs is small and stable. The computational effort increases when problems with 500 jobs are considered, but the CPU time remains acceptable and stable.

The average number of characterized sequences is usually very high, especially for large problem instances. Nevertheless, it varies according to the considered problem instance (while usually remaining high).

## 9 Conclusion

In this paper, a dominance theorem is studied which characterizes, given the interval structure of a problem, a set of dominant sequences. An interesting property of this characterized set is that both its flexibility and its performance with regard to the lateness can be computed, without any sequence enumeration, in a polynomial time. Another interesting property is that it remains unchanged as long as the relative order among the release and the due dates of the jobs is conserved. Furthermore, it is independent of the processing times. On the basis of these properties, it has been shown that, given an interval structure, a model of uncertainty can be found, associating to each parameter  $r_i$ ,  $p_i$ ,  $d_i$  an interval. Then, provided that the intervals  $[r_i]$  and  $[d_i]$  are disjoint, a worst performance can be ensured regarding the lateness of each job. The worst performance is robust both for any execution scenario with respect of the model and for any dominant sequence characterized by the theorem.

A branch and bound procedure has been also proposed which modifies the interval structure attached to a problem so that the non-optimal sequences of the dominant set are progressively discarded. The nodes of the exploration tree correspond to new problems which differ from the initial one in the fact that the release and due dates of the jobs have been respectively increased and/or decreased. Thus, any leaf of the exploration tree is a problem having an interval structure such that the dominance theorem only characterizes optimal sequences. The experimentation has shown that the procedure is able to characterize, in a reasonable time, a very impressive number of optimal sequences.

The previous results can be reused to characterize a set of solutions for job shop problems. Indeed, as a job shop problem with  $m$  machines can be divided into  $m$  single machine scheduling problems, a set of solutions can be determined by characterizing a set of sequences for each machine. Of course, these sets have to be coherent together, i.e. if  $i$  and  $j$  are two tasks, to be respectively achieved on  $M_1$  and  $M_2$ , such that  $i \prec j$ , then the worst earliest starting time of  $i$  on  $M_1$  has to be less than or equal to the best earliest starting time of  $j$  on  $M_2$ .

## References

- [1] Adams, J., Balas, E., and Zawack, D., The shifting bottleneck procedure for job shop scheduling, 1988, *Management Science*, 34(3), 391-401.
- [2] Akturk, M.S., & Gorgulu, E., Match-up scheduling under a machine breakdown, 1999, *European Journal of Operational Research*, 112, 81-97.
- [3] Allen, J., An interval based representation of temporal knowledge, 1981, *Proceedings of the 7<sup>th</sup> IJCAI*, Vancouver, Canada, 221-226.
- [4] Aloulou, M. A., Portmann, M.-C., & Vignier, A., Predictive-reactive scheduling for the single machine problem, 2002, 8<sup>th</sup> Workshop on Project Management and Scheduling, Valencia, 3-5 april, 39-42.
- [5] Artigues, C., & Roubellat, F., Characterization of a set of schedules in resource-constrained multi-project scheduling problem, 1999, *International Journal of Industrial Engineering - Theory, Applications and Practice*, 6(2), 112-122.
- [6] Bertsimas, D., & Sim, M., The price of robustness, 2004, *Operations Research*, 52, 35-53.
- [7] Billaut, J.-C., & Roubellat, F., Characterization of a set of schedules in a multiple resource context, 1996, *Journal of Decision Systems*, 5(1-2), 95-109.
- [8] Billaut, J.-C., Moukrim, A., & Sanlaville, E., Introduction à la flexibilité et à la robustesse en ordonnancement, 2004, *Flexibilité et Robustesse en Ordonnancement (in french)*.
- [9] Briand, C., Despontin, E., & Roubellat, F., Scheduling with time lags and preferences : a heuristic, 2002, 8th International Workshop on Project Management and Scheduling (PMS'02), 77-80, Valencia, Spain.
- [10] Briand, C., La, H.T., & Erschler, J., Ordonnancement de problèmes à une machine: une aide à la décision pour un compromis flexibilité vs performance, 2003, 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'03) (*in french*).
- [11] Briand, C., La, H.T., & Erschler, J., A new sufficient condition of optimality for the two-machine flowshop problems, 2006, *European Journal of Operational Research*, 169, 712-722.
- [12] Carlier, J., The one-machine sequencing problem, 1982, *European Journal of Operational Research*, 11, 42-47.
- [13] Carlier, J., & Pinson, E., An algorithm for solving the job shop problem, 1989, *Management Science*, 35(2), 164-176.
- [14] Cheng, J., Steiner, G., & Stephenson, P., Fast algorithms to minimize the makespan or maximum lateness in the two-machine flow shop with release times, 2002, *Journal of Scheduling*, 5, 71-92.

- [15] Chiang, W.-Y., & Fox, M.S., Protection against uncertainty in a deterministic schedule, 1990, Fourth International Conference on Expert Systems in Production and Operations Management, South California, USA.
- [16] Chu, C., Portmann, M.C., & Proth, J.M., A Splitting up Approach to Simplify Job-shop Scheduling Problems, 1992, International Journal of Production Research, 30(4), 859-870.
- [17] Daniels, R.L., & Carrillo, J.E.,  $\beta$ -Robust scheduling for single-machine systems with uncertain processing times, 1997, IIE Transactions, 29, 977-985.
- [18] Davenport, A.J., & Beck, J.C., A survey of techniques for scheduling with uncertainty, 2000, *unpublished*, available at <http://www.eil.utoronto.ca/profiles/chris/chris.papers.html>.
- [19] Davenport, A.J., Gefflot, C., & Beck, J.C., Slack-based Techniques for Robust Schedules, 2001, Six European Conference on Planning (ECP-2001), Toledo, Spain.
- [20] 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, 1983, Operations Research, 31, 114-127.
- [21] Esquirol, P., Lopez, P., & Mancel, P., Représentation et traitement du temps en ordonnancement, 1999, Rapport LAAS N°99455 (in french).
- [22] Esswein, C., & Billaut, J.-C., Trade-off between flexibility and maximum completion time in the two-machine flowshop scheduling problem, 2002, International Symposium on Combinatorial Optimisation, Paris (France).
- [23] Esswein, C., Un apport de flexibilité séquentielle pour l'ordonnancement robuste, 2003, PhD Thesis, Université de Tours, France (*in french*).
- [24] Esswein, C., Billaut, J.-C., & Artigues, C., Une approche multi-critères pour un apport de flexibilité séquentielle, 2004, Flexibilité et robustesse en ordonnancement (*in french*).
- [25] Gao, H., Fox, M.S., Chiang, W.-Y., & Hikita, S., Building Robust Schedules - An Empirical Study of Single Machine Scheduling with Uncertainty, 1995, unpublished paper.
- [26] GOTHa, Flexibilité et Robustesse en Ordonnancement, 2002, Le bulletin de la ROADEF (*in french*).
- [27] Hariri, A.M., & Potts, C.N., An algorithm for single machine sequencing with release dates to minimize total weighted completion time", 1983, Discrete Applied Mathematics, 5, 99-109.
- [28] Herroelen, W., & Leus, R., Project scheduling under uncertainty - survey and research potentials, 2003, European Journal of Operational Research, *to appear*.
- [29] Herroelen, W., & Leus, R., Robust and reactive project scheduling - a review and classification of procedures, 2004, International Journal of Production Research, vol. 42, 8, 1599-1620.

- [30] Herroelen, W., & Leus, R., The construction of stable project baseline schedules, 2004, *European Journal of Operational Research*, 156, 550-565.
- [31] Jensen, M.T., *Robust and Flexible Scheduling with Evolutionary Computation*, 2001, PhD Thesis, Department of Computer Science, University of Aarhus, Denmark.
- [32] Kouvelis, P., Daniels, R.L., & Vairaktarakis, G., Robust scheduling of a two-machine flow shop with uncertain processing times, 2000, *IIE Transactions*, 32, 421-432.
- [33] Le Gall, A., *Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'ateliers*, 1989, PhD Thesis, Université Paul Sabatier, Toulouse, France (*in french*).
- [34] Leon, V.J., Wu, S.D., & Storer, R.H., Robustness measures and robust scheduling for job shops, 1994, *IIE Transactions*, 26, 32-43.
- [35] Mauguière, P., Billaut, J.-C., & Artigues, C., Grouping on a single machine with heads and tails to represent a family of dominant schedules, 2002, 8<sup>th</sup> Workshop on Project Management and Scheduling, Valencia, 3-5 April, 265-269.
- [36] Mehta, S.V., & Uzsoy, R.H., Predictable Scheduling of a Job Shop Subject to Breakdowns, 1998, *IEEE Transactions on Robotics and Automation*, 14, 365-378.
- [37] Moukrim, A., Sanlaville, E., & Guinand, F., Parallel machine scheduling with uncertain communication delays, 2003, *RAIRO Operations Research*, 37, 1-16.
- [38] Sabuncuoglu, I., & Bayiz, M., Analyse of reactive scheduling problems in a job shop environment, 2000, *European Journal of Operational Research*, 126, 567-586.
- [39] El Sakkout., Richards, E.T., & Wallace, M.G., Unimodular Probing for Minimal Perturbance in Dynamic Resource Feasibility Problems, 1997, CP97 Workshop on the Theory and Practice of Dynamic Constraint Satisfaction.
- [40] Snoek, M., Anticipation Optimization in Dynamic Job Shops, 2001, Genetic and Evolutionary Computation Conference 2001 (GECCO-2001), San Francisco, 43-46.
- [41] Smith, S.F., *Reactive scheduling systems*, 1995, *Intelligent Scheduling Systems*, Brown, D.E., & Scherer, W.T, Kluwer Press.
- [42] Tavares, L.V., & Ferreira, J.A.A., & Coelho, J.S., On the optimal management of project risk, 1998, *European Journal of Operational Research*, 107, 451-469.
- [43] Thomas, V., *Aide à la décision pour l'ordonnancement d'ateliers en temps réel*, 1980, PhD Thesis, Université Paul Sabatier, Toulouse, France (*in french*).
- [44] Wiers, W.C.S., A review of the applicability of OR and AI scheduling techniques in practice, 1997, *Omega-International Journal of Management Science*, 25, 145-153.

- [45] Wu, D., Byeon, E.S., & Storer, R.H., A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness, 1999, *Operations Research*, 1, 113-124.