A Cooperative Approach for Job Shop Scheduling under Uncertainties

C. BRIAND[a,1], S. OURARI[b] and B. BOUZOUIAI[b]
[a] *Université de Toulouse, LAAS CNRS, France*
[b] *CDTA, Alger, Algérie*

Abstract. This paper focuses on job shop scheduling problems in a cooperative environment. Unlike classical deterministic approaches, we assume that jobs are not known in advance but occur randomly during the production process, as orders appear. Therefore, the production schedule is adapted in a reactive manner all along the production process. These schedule adaptations are made according to a cooperative approach, that is the major originality of this paper. Each resource manages its own local schedule and the global schedule is obtained by point-to-point negotiations between the various machines. We also suppose that local schedules are flexible since several alternative job sequences are allowed on each machine. This flexibility is the key feature that allows each resource, on the one hand, to negotiate with the others and, on the other hand, to react to unexpected events. The cooperative approach aims at ensuring the coherence between the local schedules while keeping a given level of flexibility on each resource.

Keywords. cooperative scheduling, flexibility, robustness, dominance.

Introduction

Many research efforts in scheduling assume a static deterministic environment within which the schedule is executed. However, considering any real enterprise environment, the probability for a pre-computed predictive schedule to be executed as planed is quite weak. Many parameters related to a scheduling problem are in fact subject to fluctuations. The disruptions may arise from a number of possible sources [3][7][13]: job release dates and job due dates may change, new jobs may need to be taken into account, operation processing times can vary, machine can breakdown, etc.

One way to take theses disruptions into account, while keeping the schedule performance under control, consists to use a two-level resolution scheme: an off-line scheduling level builds up a predictive schedule, then a on-line scheduling level adapts the predictive schedule all along the production process, taking disruptions into account. Notions of flexibility and robustness are defined in order to characterize a scheduling system able to resist to some parameter variations. The flexibility refers to the fact that some schedule decisions are kept free during the off-line phase in order to be able to face unforeseen disturbances during the on-line phase [12]. Robustness is closely linked to flexibility; actually, flexibility is often injected into a deterministic solution in order to make it robust with regard to some kinds of uncertainty sources.

[1] Corresponding Author: C. Briand, Université de Toulouse, LAAS CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse, Email: briand@laas.fr

The literature connected to scheduling with uncertainties is growing. A classification of the scheduling methods is proposed in [7], [17] and [13]. Some methods generate a predictive schedule which is reactively repaired (i.e. some local adaptations are made inside the schedule) for taking unexpected events into account, aiming at minimizing the perturbation made inside the original schedule [21]. Others approaches, referred to as proactive, construct predictive schedules on the basis of a statistical knowledge of the uncertainties, aiming at determining a schedule having a good average performance [16][22][23]. Some other approaches use both proactive and reactive methods since all unexpected events cannot be taken into account inside the proactive phase. In this class of approaches, a temporal flexibility [13][16] or a sequential flexibility [2][5][11] is often 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 moving from an obsolete solution to another one, while minimizing the performance loss. Among such approaches, one can distinguish approaches based on the notion of operation groups [2][3][11] which allow the permutation of certain contiguous tasks on a resource. Another kind of approach is proposed in [5] for the one machine problem, where the characterization of a flexible family of solutions is based on the use of a dominance theorem. This approach is used in this paper.

Basically, in the on-line as well as in the off-line phases, scheduling is usually considered as a global decision problem since the scheduling decisions deal with the organization of all the resources [6]. However, in many application domains (supply chain management, industrial projects, timetabling …), resources are often distributed among a set of actors which get their own decisional autonomy. Consequently, a global scheduling approach seems unrealistic since each actor cannot control its own organization. In this case, a cooperative approach is better suited: scheduling decisions have to be negotiated between the actors intending to converge towards a compromise that satisfies both local and global performances. Such approaches are proposed in [8][18][19] [20].

The paper is organised as follows: at first, some notions, useful for the comprehension of the proposed approach, are described in Section 1. Section 2 presents the assumptions that have been made for defining the cooperative scheduling problem. Section 3 introduces the global coherence notion as well as the various cooperation functions. Section 4 focuses on the negotiation process that concerns pair of actors, this negotiation process being more formalized in Section 5.

1. A Robust Approach for the Single Machine Scheduling Problem

A single machine problem consists of a set V of n jobs to be scheduled on a single disjunctive resource. The processing time p_j, the release date r_j and due date d_j of each job j are known. The interval $[r_j\ d_j\ ]$ defines the execution window of each job j. A job sequence is referred to as feasible if all the jobs of V are completed early, i.e. if Eqs (1) is satisfied. Regarding the feasibility objective, this problem is NP-hard [15].

$$\forall\, i \in V, \quad s_i \geq r_i \quad et \quad f_i = s_i + p_i \leq d_i \tag{1}$$

Considering the one machine scheduling problem with execution windows, a dominance theorem is stated in the early eighties by Erschler et al. [9]. The theorem uses the notions of top and pyramid which are defined on the basis of the job execution intervals [r_i d_i]. Let us remind to the reader that a condition of dominance enables the reduction of the solution search space: only the non-dominated solutions are kept. We notice that for a one machine problem, a sequence S_2 is dominated by another sequence S_1 if the feasibility of S_2 implies that S_1 is feasible. Before presenting the theorem, the notions of a top and a pyramid need to be defined.

Definition 1. A job $t \in V$ is called a top if there does not exist any other job $i \in V$ such that $r_i > r_t \wedge d_i < d_t$

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.

Definition 2. Given a top t_α, a pyramid P_α related to t_α is the set of jobs $i \in V$ such that $r_i < r_{t_\alpha} \wedge d_i > d_{t_\alpha}$

Considering Definition 2, it can be noticed that a non-top job may belong to several pyramids. The functions *u(j)* and *v(j)* indicate the index of the first pyramid to which the job *j* belongs and the index of the last job to which the job *j* belongs, respectively. Erschler et al. give the proof of the following theorem, further referred to as pyramidal theorem, in [9].

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 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 equality);

- only the jobs belonging to the 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 both to 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).*

The previous theorem enables to characterise a set of dominant sequences. Let us note that this set of dominant sequences is independent of the numerical values of the processing times p_j as well as the explicit values of r_i and d_i. Only the total relative order of the release and due dates is considered. In [10] and [4], it is shown that this set

is also dominant with regards to the optimisation of the regular criteria T_{max} , the maximum tardiness, and L_{max}, the maximum lateness.

In [5], it is also shown how, given a problem V and its corresponding set of dominant sequences S^V, determined in accordance with the pyramid theorem; it is possible to associate to each job i a lateness interval $[L_i^{min}, L_i^{max}]$ where L_i^{min} and L_i^{max} respectively represent the best and the worst lateness of job i among all sequences of S^V.

The computation of the lateness intervals can be performed in polynomial time by determining for each job j the most unfavorable and the most favorable sequences, i.e. the sequences implying the smallest and greatest delays for the job amongst all sequences in S^V respectively. Figure 1 depicts the structure of these sequences for any job j. The notations A, B and σ represent job sub-sequences as illustrated in Figure 1, and t_k is the k^{th} top.

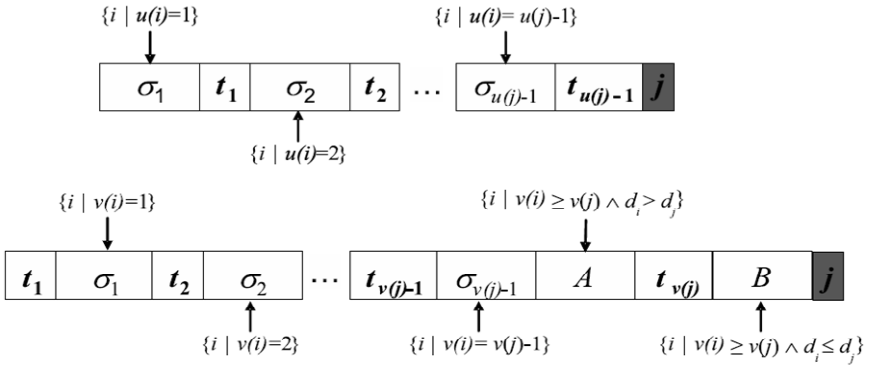

Figure 1. Most favorable and unfavorable sequences for a job j

Given $L_i^{\min}$ and $L_i^{\max}$, the optimal lateness $L^{\max}$ is bounded as follows:

$$\underset{i\in V}{Max}\left(L_i^{\min}\right) \le L_{\max} \le \underset{i\in V}{Max}\left(L_i^{\max}\right) \tag{2}$$

According to Eq. (2), it is possible to determine whether a dominant set of sequences is acceptable, relatively to its worst performance. On the other hand, it is shown in [5] and [14] how to eliminate from the dominant set some "worst sequences" in order to enhance the worst performance.

Let us also remark that the L_i^{min} and L_i^{max} values allow to deduce the values of the best and worst starting time s_i^{min} and s_i^{max} of each job i, according to Eq. (3).

$$s_i^{\min} = L_i^{\min} + d_i - p_i \text{ and } s_i^{\max} = L_i^{\max} + d_i - p_i \tag{3}$$

2. Cooperative Framework

The Job Shop Scheduling problem is considered as follows. T jobs have to be processed on $M=\{M_1,...M_m\}$ machines. Jobs consist of a sequence of operations to be carried out on the machines according to a given routing. Each machine can process

only one operation at a time. The j^{th} operation of a job i is referred to as o_{ij}. Its duration id denoted $p_{i,j}$. The commonly considered objective is to minimize the makespan which corresponds to the total duration of the schedule. The job shop scheduling problem is NP-hard [15]. It is possible to decompose it into m interdependent one machine sub-problems, where each operation is characterized by its execution interval $[r_{ij}, d_{ij}]$. For each sub-problem, the objective consists to minimize the maximum lateness. We highlight that the sub-problems are interdependent because the optimal sequence determined on a given machine must be consistent (according to the routing constraints) with the other optimal sequences established for the other resources [1].

As stated above, we suppose that each resource is associated with a decision center (DC) which manages its own local schedule, exchanging information and products with other DCs. We also assume that each DC gets its own decisional flexibility. In our case, this flexibility corresponds to the number of dominant sequences that can be characterized by the pyramidal theorem.

The DCs are crossed by products flows. Considering a particular DC_i, we can distinguish, according to the routing constraints, its upstream centers and its downstream centers. An upstream center is the supplier of the downstream center, which can be viewed in its turn as its customer.

In this work, we assume that each DC cooperates with its upstream and downstream DCs using point-to-point communication. The cooperation aims at bringing the actors to collectively define (by negotiation) the delivery date of the products, while giving each actor enough flexibility for reacting to disturbances.

We now consider a negotiation process carried out between two DC*s*. If we focus on decision center DC_i which must perform a set of operations V_i, this DC has to negotiate for each operation $o_{u,v} \in V_i$:

- with the upstream DC which performs $o_{u,v-1}$, in order to define a temporal interval $[\, r_{uv}^{\min} \;\; r_{uv}^{\max} \,]$ (further referred to as $[r_{uv}]$) corresponding to the availability interval of $o_{u,v-1}$.
- with the downstream DC which performs $o_{u,v+1}$, in order to define a temporal interval $[\, d_{uv}^{\min} \;\; d_{uv}^{\max} \,]$ (further referred to as $[d_{uv}]$) corresponding to the delivery interval of $o_{u,v}$.

We note that interval $[r_{uv}]$ for the machine performing o_{uv} corresponds to interval $[d_{u,v-1}]$ for the machine performing $o_{u,v-1}$. Also, $[d_{uv}] = [r_{u,v+1}]$.

We also highlight that setting product delivery intervals between two DCs (instead of a fixed delivery date) gives more flexibility to each DC for managing its own local organization. We assume that the $[r_{uv}]$ and $[d_{uv}]$ negotiations give rise to contracts between DCs. This contract corresponds to a supplier-customer mutual commitment: the upstream DC commits to delivery its product in a given temporal interval, while the downstream DC commits to start the execution of the next operation on this product in the same interval.

3. Local and Global Consistency and Cooperation Functions

Under the previously defined assumptions, each DC has to build up a local schedule that must be consistent with the availability and delivery windows $[r_{uv}]$ and $[d_{uv}]$. Using the approach described in Section 1, each local schedule is flexible, and each operation

is characterized by a best and a worst starting date [$s_{uv}^{\min} s_{uv}^{\max}$] (referred to as [s_{uv}]). The local consistency of [s_{uv}] with the negotiated intervals [r_{uv}] and [d_{uv}] can be expressed by the following inequalities:

$$r_{uv}^{\min} \le s_{uv}^{\min} \le r_{uv}^{\max} \text{ and } d_{uv}^{\min} \le s_{uv}^{\max} + p_{uv} \le d_{uv}^{\max} \quad (3)$$

The above conditions (2) impose, on the one hand, that a DC never plans to start the execution of an operation of a job before it becomes available (i.e. $r_{uv}^{\min} \le s_{uv}^{\min}$) and, on the other hand that, in the worst case, the job must be delivered on time (i.e. $s_{uv}^{\max} + p_{uv} \le d_{uv}^{\max}$). The others two inequalities avoid a over-autonomy state, in which a DC would ask an upstream DC to achieve a job earlier than necessary (i.e. $r_{uv}^{\max} < s_{uv}^{\min}$), and the situation where a job would be achieved, in the worst case, earlier than necessary (i.e. $s_{uv}^{\max} + p_{uv} < d_{uv}^{\min}$).

As in [18], we consider that the underlying functions of a cooperation process are the negotiation, the coordination and the renegotiation. A negotiation process is initiated when a DC asks for the first time its upstream or downstream DCs to perform an operation on a new job to be taken into account in the shop. This operation corresponds to the next or to the preceding operation to be performed on the job, according to its routing. Negotiation aims at finding the intervals [r_{uv}] or [d_{uv}] for operation o_{uv}. Let us remark that new job arrival corresponds to the occurrence of a new order. We suppose that a delivery interval is associated to the order (this interval being eventually reduced to a point). The aim of the negotiation is to define the intervals [r_{uv}] and [d_{uv}] of the new operation, trying to respect the local consistency constraints (see Eq. (3)) for the already existing operations.

During the negotiation related to the insertion of a new operation, it can be suitable to renegotiate some already existing intervals [r_{ij}] and [d_{ij}] so as to improve the completion time of the new operation. This renegotiation situation also occurs when a disturbance makes an interval [s_{ij}] inconsistent (with regards to Eq (3)) with the current values of [r_{ij}] and [d_{ij}]. The goal of the renegotiation process is to recover the local consistency.

Negotiation and renegotiation are carried out by exchanging interval requests between pairs of DCs. An initiator DC issues an interval proposal [r_{uv}] or [d_{uv}] to another either upstream or downstream DC. The latter can either accept the proposal or refuse it by issuing a counterproposal.

Since local schedules are evolving over time, it is necessary that DCs coordinate themselves so that the organization remains globally feasible (i.e. Eq. (4) must be satisfied). This coordination corresponds to the asynchronous exchange of the values of the availability and delivery intervals of the operations. When Condition (4) is violated, a re-schedule has to be performed on the DC which detects the violation. Moreover, if the change of a best or worst start date is inconsistent with Condition (3) then a renegotiation is imposed between the concerned DCs.

$$s_{uv}^{\min} \ge f_{u,v-1}^{\min} = s_{u,v-1}^{\min} + p_{u,v-1} \text{ and } s_{uv}^{\max} \ge f_{u,v-1}^{\max} = s_{u,v-1}^{\max} + p_{u,v-1} \quad (4)$$

We point out that, during the various negotiations and renegotiations, issuing consistent proposals and counterproposals is not trivial. The following section focuses on this aspect.

4. Negotiation and Renegotiation

Considering a DC, the determination of a set of dominant sequences requires to define a total order among the r_{uv} and d_{uv} of all the jobs that the DC performs. Indeed, this order is required in order to apply the pyramidal theorem (see Section 1). Let us highlight that, while the negotiation process allows to determine the intervals $[r_{uv}]$ and $[d_{uv}]$, the values of r_{uv} and d_{uv} are not precisely fixed, hence there exists several possible total orders. Moreover, for each considered total order corresponds a specific dominant set of sequences, hence different values of $[\, s_{uv}^{\min} s_{uv}^{\max} \,]$. The determination on a DC of a pertinent total order between r_{uv} and d_{uv} requires to compare interval $[s_{uv}]$ with $[f_{u,v-1}]$ and $[f_{uv}]$ with $[s_{u,v-1}]$. These comparisons lead us to define an *inconsistency risk* notion.

We say that an inconsistency risk exists between two operations o_{ij} and $o_{i,j+1}$ if $s_{i,j+1}^{\min} < f_{ij}^{\max}$ is valid, i.e. the intervals $[f_{ij}]$ and $[s_{i,j+1}]$ overlap. Ideally, when the intervals $[f_{ij}]$ and $[s_{i,j+1}]$ do not overlap (see Figure 2), the inconsistency risk is null. Indeed, in this optimistic case, the worst completion value of o_{ij} is always consistent with the best earliest start date of $o_{i,j+1}$ whatever the execution sequences of the jobs will be on each DC. Nevertheless, in the general case, overlapping is allowed (see Figure 3). In this case there is a risk for the completion date of o_{ij} to be greater than the start date of $o_{i,j+1}$. Obviously, the larger the overlap interval, the greater the inconsistency risk.

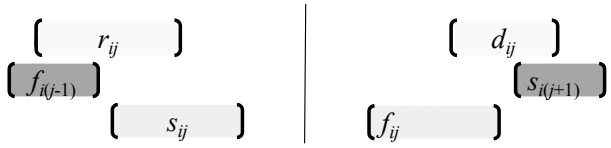

Figure 2. Case of a null inconsistency risk

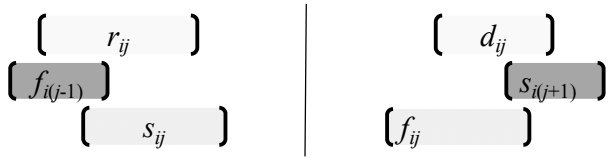

Figure 3. General case

The determination of a total order between release date and due dates must be guided by the aim of minimizing the inconsistency risk. However, this goal is not the only one to be considered. As discussed above, it is also necessary to maximize the DC's decisional flexibility so that the DC keeps its ability to face production disturbances or new job arrivals.

As a matter of fact, the decisional flexibility is linked to the number of dominant sequences characterized by each DC. An interesting property of the pyramidal theorem is that this number can be easily computed, using Eq. 5, without requiring any sequence

enumeration. In this formula, n_q is the number of non-top jobs exactly belonging to q pyramids and N, the total number of pyramids.

$$\pi = \prod_{q=1}^{N} (q+1)^{n_q} \tag{5}$$

We can notice that π is maximal when there exist only one pyramid containing all the jobs. In this case π equals 2^{n-1}, where n is the number of jobs that the DC manages. Also we note that π decreases as pyramid number increases. In the worst case, all operations are tops, and $\pi = 1$.

Minimizing the inconsistency risk and maximizing the flexibility are two contradictory objectives. Indeed, the greater the number of dominant sequences, the wider the intervals $[s_{uv}]$, and subsequently, the greater the inconsistency risk. Conversely, to reduce the inconsistency risk, it is necessary to lose flexibility in order to tight the interval widths. Let us highlight that the above trade-off is also connected to the global system performance. Indeed, regarding a job shop scheduling problem with makespan minimization as global objective, the performance of the global schedule is much more effective as interval $[s_{uv}]$ decreases and flexibility decreases. Indeed, in every optimal schedule, any interval $[s_{uv}]$ is reduced to a point and the DC's flexibility is null.

5. A First Approach for Inter-DC Negotiation

In the previous section, we put in evidence the need of making a trade-off between inconsistency risk minimization and flexibility maximization. In the remainder of the paper, a first negotiation approach is sketched which enables to formalize a DC behaviour.

To simplify the problem, it is assumed that the pyramidal structure associated to each DC is such that each operation belongs to a single pyramid (i.e. $u(o_{ij})=v(o_{ij})$). Under this assumption, pyramids are referred to as independent. Therefore, a best completion time $f_P^{\min}$ and a worst completion time $f_P^{\max}$ can be associated to each pyramid P.

When a new operation o_{xy} has to be considered by a DC, it systematically receives a proposal from another upstream or downstream DC. If the proposal comes from an upstream center, a target interval of availability date $[r_{xy}]$ is proposed, in the other case, a target interval of delivery time $[d_{xy}]$ is proposed. Let us note that, independently from the proposal origin, it is always possible by propagation of the precedence constraints to determine a pair of target intervals $[r_{xy},][\ d_{xy}]$.

When receiving a negotiation proposal, a DC must at first, decides to which pyramid the new operation o_{xy} will be assigned: for instance, either to an already existing pyramid or to a new pyramid (of top o_{xy}) which will be inserted between two existing pyramids. The assignment of a new operation inside a pyramidal structure defines an optimization problem which is not addressed in this paper. Let us remark that this problem can be very complex when pyramid splitting is allowed. We also point out that the flexibility of the DC entirely depends on the solution of this problem since the flexibility is inversely proportional to the pyramid number.

Now, assuming that the new-task assignment has been decided (i.e. $o_{xy} \in P_\alpha$), a total order among the availability and due dates of the operations belonging to P_α must be defined. Indeed, this total order is required in order to determine the interval values $[s_{uv}]$ for each $o_{uv} \in P_\alpha$. This is also an optimisation problem where the objective is, at this time, to minimize the inconsistency risk.

In order to formalize this problem, a linear program is proposed below. Without loss of generality, we suppose that the n_α jobs of pyramid P_α are indexed according to the increasing order of $r_{ij}^{\min}$. For simplification, we also suppose that the order of the availability dates of the operations of P_α match the increasing order of $r_{ij}^{\min}$ (this assumption tends to favour the coherence between the $r_{ij}^{\min}$ and $s_{ij}^{\min}$ values). Since there is only one top in P_α, one consequence of this assumption is that the operation having the index n_α defines the top of P_α because this operation has the greatest availability date. Now it only remains to determine the order of the due dates and the intervals values $[s_{uv}]$ for each operation $o_{uv} \in P_\alpha$.

For this purpose, a linear program (LP) with integer variables is proposed below. The main decision variables are x_{ij}, $s_{ix}^{\min}$ and $s_{ix}^{\max}$. The decision variables x_{ij} are binary: x_{ij}=1 if the deadline of i is greater than the one of j, else x_{ij}=0. The values of these variables allow to deduce the total order of the due dates of the operations of P_α. Variables $s_{ix}^{\min}$ and $s_{ix}^{\max}$ are integers, and correspond respectively to the best and the worst start dates of o_{ix}. The parameters of the LP are the values of intervals $[r_{ix}]$ and $[d_{ix}]$, the processing time p_{ix} and the weights $w_{i(x-1)}$ and $w_{i(x+1)}$. The significance of these weights is further discussed in the paper.

$$\min L$$

$$x_{ij} + x_{ji} = 1 \qquad \forall (i,j), i \neq j \qquad (6)$$

$$x_{ij} + x_{jk} - x_{ii} \leq 1 \qquad \forall (i,j,k) \qquad (7)$$

$$x_{ii} = 0 \qquad \forall i \qquad (8)$$

$$x_{in_\alpha} = 1 \qquad \forall i, i \neq n_\alpha \qquad (9)$$

$$s_{ix}^{\min} \geq s_{jy}^{\min} \geq f_{P_{\alpha-1}}^{\min} \qquad \forall (i,j), i > j \qquad (10)$$

$$s_{ix}^{\max} \geq \max(s_{jy}^{\min}, f_{P_{\alpha-1}}^{\max}) + \sum_{k=j}^{n_\alpha} x_{ki}\, p_{ki} + \sum_{k=1}^{n_\alpha} x_{ik}\, p_{ik} \qquad \forall (i,j), i \neq j \qquad (11)$$

$$L \geq w_{i(x-1)} (r_{ix}^{\max} - s_{ix}^{\min}) \qquad \forall i \qquad (12)$$

$$L \geq w_{i(x-1)} (s_{ix}^{\min} - r_{ix}^{\max}) \qquad \forall i \qquad (13)$$

$$L \geq w_{i(x+1)} (s_{ix}^{\max} + p_{ix} - d_{ix}^{\min}) \qquad \forall i \qquad (14)$$

$$L \geq w_{i(x+1)} (d_{ix}^{\min} - s_{ix}^{\max} - p_{ix}) \qquad \forall i \qquad (15)$$

Constraints (6)–(8) ensure that deadlines are totally ordered. Constraint (7) states that $(x_{ij} = 1) \wedge (x_{jk} = 1) \Rightarrow (x_{ik} = 1)$. Constraint (9) imposes to the operation associated

to the job n_α to be the top (it gets the smallest due date). Constraint (10) ensures that the best and the worst start dates respect the increasing order of $r_{ij}^{\min}$. We also impose, according to the most favourable sequences (see figure 1), that these dates must be greater than the best completion date of the preceding pyramid $P_{\alpha-1}$. Constraint (11) imposes that the worst completion date of each operation is determined according to the structure of the most unfavourable sequences (see figure 1). The expression $\sum_{k=j}^{n_\alpha} x_{ki}\, p_k$ corresponds to the sum of the processing time of the jobs belonging to set A in the unfavourable sequence, while $\sum_{k=1}^{n_\alpha} x_{ik}\, p_k$ corresponds to that of jobs belonging to set B.

The LP objective is the minimisation of the variable L which is determined according to constraints (12)-(15). This variable measures the greatest gap between $s_{ix}^{\min}$ and $r_{ix}^{\max}$ on the one hand, and $s_{ix}^{\max} + p_{ix}$ and $d_{ix}^{\min}$ on the other hand. In other words, we search to minimize the advance-delay of the operations according to contracted intervals $[r_{ix}]$ and $[d_{ix}]$. When $L = 0$, the inconsistency risk is obviously null, since the best start date of o_{uv} is equal to the worst delivery date of $o_{u,v-1}$ and the worst completion time of o_{uv} is equal to the best delivery date of $o_{u,v+1}$.

The weights $w_{i(x-1)}$ and $w_{i(x+1)}$ ponder, for each operation, the advance-delay according to the upstream/ downstream resource priority. The idea is to penalize more an advance-delay concerning an upstream/downstream resource having poor flexibility, and vice-versa.

The optimal solution of this LP allows the DC to know for all the managed operations, the values of their $s_{uv}^{\min}$ et $s_{uv}^{\max}$, and in particular, those of $s_{xy}^{\min}$ and $s_{xy}^{\max}$. On the basis of these values, negotiation/renegotiation proposals issued by the DC can be elaborated.

Conclusion

In this paper, the job shop scheduling problem under uncertainties is considered. At the opposite of the classical centralised scheduling techniques, a cooperative approach which gives each resource a decisional autonomy is proposed. The scheduling decisions result from point-to-point negotiations and renegotiations between resources. The cooperation process aims at characterizing a flexible family of solutions on each resource, this flexibility being used for facing production disruptions and new job arrivals. A negotiation/renegotiation process is started when a new operation is inserted, or when unexpected event occurs. It leads to adapt the interval structure associated to each resource. Basically, if the sequential flexibility is large, then the ability to absorb unexpected events is important, but the inconsistency risk becomes greater and the global performance gets weaker. Reversely, when the flexibility gets weaker, the global performance increases, however it is less reliable. Several cooperation bases have been laid down and a first formalization of the negotiation/renegotiation processes has been proposed. Our mid-term goal is to implement an automatic cooperative scheduling prototype in order to validate the cooperation approach and to improve it.

References

[1] J.Adams, E.BAlas & D.Zawack. The shifting bottleneck procedure for job shop scheduling. Management Science, vol.34, n°3, pages 391-401, 1988.

[2] A. Aloulou & M-C Portmann, Définition d'ordonnancements flexibles. Première application à un problème à une machine. GI'2001, Aix-en-Provence, 2001.

[3] Billaut J.C., Moukrim A., & Sanlaville E., Flexibilité et robustesse en ordonnancement. Hermes, Traité IC2, ISBN 2-7462-1028-2, 2005.

[4] C.Briand, M-J.Huguet, H.T.La & P.Lopez. Approche par contraintes pour l'ordonnancement robuste. Dans J-C.Billaut, A.Moukrim & E.Sanlaville, editeurs, Flexibilité et Robustesse en Ordonnancement, Traité IC2, ISBN 2-7462-1028-2, 2005, pp. 191-215.

[5] C. Briand, H. Trung La, & Jacques Erschler, A robust approach for the single machine scheduling problem. Journal of Scheduling, Special Issue on Project Scheduling under Uncertainty, Demeulemeester, E.L. and Herroelen W.S. (eds)., vol. 10, no. 3, pp 209-221, 2007.

[6] Chu C. & Proth J.M., L'ordonnancement et ses applications. Masson, Paris, France, 1996.

[7] Davenport A.J. & Beck J.C., A survey of techniques for scheduling with uncertainty. Disponible en ligne, 2000.

[8] Dudek G., Stadtler H., Negotiation-based collaborative planning between supply chain partners, European Journal of Operational Research, vol. 163, pp 668-687, 2004.

[9] J. Erschler, G. Fontan, C. Merce & F. Roubellat, A new dominance concept in scheduling n jobs on a single machine with ready times and due dates. Operations Research, vol.1, pp. 114–127, 1983.

[10] Erschler J., Fontan G. & Merce C., Un nouveau concept de dominance pour l'ordonnancement de travaux sur une machine. RAIRO Recherche Opérationnelle/Operations Research, Vol. 19, n°1, 1985.

[11] C.Esswein, A.Puret, J.Moreira & J.C. Billaut, An efficient methode for job shop scheduling with sequential flexibility. In ORP3, 2003, Germany

[12] GOThA, Flexibilité et robustesse en ordonnancement. Le bulletin de la ROADEF, 8 :10-12, 2002.

[13] Herrolelen W. &t Leus R., Robust and reactive project scheduling : A review and classification of procedures. International Journal of Production Research, Vol.42, No.8, 1599-1620, 2004.

[14] La H.T., Santamaria J.-L. & Briand C., Une aide à la décision pour l'ordonnancement robuste en contexte mono-ressource : un compromis flexibilité/performance. 6ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'05), Tours (France), 14-16 Février 2005, pp 101-116

[15] Lenstra , J.K., Rinnooy Kan A.II.G & Brucker P., Complexity of machine scheduling problems. Annals of Discrete Mathematics 1, 343-362, 1977.

[16] Leon V.J., Wu S.D., & Store R.H., Robustness measures and robust scheduling for job shops. IIE Transactions, 26(5) : 32-43, 1994.

[17] Leus Roel, The generation of stable projects plans: complexity and exact algorithms. Thèse de doctorat, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium, 2003.

[18] E. Monsarrat, C.Briand & P. Esquirol, Aide à la décision pour la coopération interentreprise. Journal Européen des Systèmes Automatisés, 39/2005.

[19] Portmann M.-C, Mouloua Z., A window time negotiation approach at the scheduling level inside supply chains, 3rd Multidisciplinary International Conference on Scheduling: Theory and Application, MISTA'07, Paris, 28-31 august, pp410-417, 2007.

[20] Quéré Y.L., Sevaux M., Tahon C. & Trenteseaux D., Reactive scheduling of complex system maintenance in a cooperative environment with communication time. IEEE Transaction on System, Man and Cybernetic- Part C: Applications and reviews, Vol 33, No.2, May 2003.

[21] Sabuncuoglu I., & de Bayiz M., Analysis of reactive scheduling problem in a job shop environement. European Journal of Operational Research, 126 (3) 567-586, 2000.

[22] Sevaux M. & Sörensen K., A genetic algorithm for robust schedules in a just-in-time environment. Rapport de recherche, University of valenciennes, LAMIH/SP-2003-1, 2002

[23] Wu S.D, Store R.H., & Chang P.C., One-machine rescheduling heuristics with efficiency and stability as criteria. Computer and Operations Research, 20, 1-14, 1993.