Characterization of all rho-approximated sequences for some scheduling problems

Jean-Charles Billaut, Pierre Lopez

Characterization of all ρ-approximated sequences for some scheduling problems

Jean-Charles BILLAUT [1,2]
[1] Université François-Rabelais Tours
Laboratoire d'Informatique
64 avenue Jean Portalis, 37200 Tours, France
jean-charles.billaut@univ-tours.fr

Pierre LOPEZ
[2] CNRS ; LAAS ; 7 avenue du colonel Roche, 31077 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE, UT1, UTM ; LAAS ; 31077 Toulouse, France
lopez@laas.fr

Abstract

Some scheduling problems present the peculiarity to be solvable in polynomial time and to have a huge number of optimal solutions. In the disturbed environment of a production manufacturing system, where the forecasted schedule is going to change because of unexpected events or uncertainties, it can be interesting not only to know one or several optimal sequences, but the characteristics of 'good' sequences. In this paper, we focus on the characterization of all the ρ-approximated sequences, which are solutions of a scheduling problem with a performance not worse than a given distance from the value of the optimal solution.

With the support of the lattice of permutations, we define the characteristics of the optimal sequences for some particular scheduling problems. We present a method which is able, for some specific scheduling problems, to give the characteristics of all the ρ-approximated sequences. A computational experience is carried out to evaluate the performance of the proposed method.

1. Introduction

The context of production manufacturing systems is a disturbed environment, mainly for two reasons: the uncertainty of the data and the occurrence of unexpected events. It happens frequently that the schedule, which was forecasted, becomes quickly infeasible. This is the reason why several dynamic or real-time approaches, that are able to react in real time to the disturbances, have been developed during the two past decades [4, 12, 13, 14].

Among the possible dynamic approaches, some of them are based on the *characterization* of solutions [7, 8, 4, 2, 1]. Characterization means that several solutions are implicitly proposed, without enumeration, instead of only one, and a system is designed to exploit this flexibility for reacting in real time. Generally, the set of solutions which are characterized, satisfies some feasibility conditions like the respect of due dates or a bounded tardiness.

We propose in this paper a totally new method, which is able to give the characteristics of the exact set of optimal or ρ-approximated solutions regarding a given criterion. The characterization of solutions is rarely studied in literature while it can be inserted into a general method, which can help, in real time, to take a scheduling decision among those which guaranty to keep the quality of the solution.

The rest of the paper is organized as follows. Section 2 introduces the lattice of permutations associated with some basic properties and their link to scheduling problems. In Section 3, the general solving method is presented and some new scheduling problems are identified. Section 4 focuses on the modeling issue of the method by using mixed integer linear programming. It is illustrated by the application to some specific scheduling problems. Section 5 presents the computational results.

2. The lattice of permutations

2.1 Definition

Let consider the set $\{1, 2, ..., n\}$ of integers and S_n the group of all permutations on $\{1, 2, ..., n\}$. The members of S_n can be represented by strings of integers. For example if $n = 4$, $\sigma = 4123$ denotes a permutation σ where $\sigma(1) = 4$, $\sigma(2) = 1$, $\sigma(3) = 2$, and $\sigma(4) = 3$. We define $index(i, \sigma)$ [11] as the position of i in σ ($index(3, \sigma) = 4$ in the example). For more simplicity, we will note $i \prec_\sigma j$ if "i precedes j in σ", or $index(i, \sigma) < index(j, \sigma)$.

With the elements of S_n, we can build a graph where there is an edge between two nodes σ and σ' if $\exists i$ and j, $i < j$, such that $index(j, \sigma) = index(i, \sigma) + 1$ and

$index(i, \sigma') = index(j, \sigma') + 1$, i.e. i immediately precedes j in σ and j immediately precedes i in σ'. This graph is a lattice: a partially ordered set of nodes in which every two nodes have a least upper bound and a greatest lower node. The lattice of permutations (also called *permutohedron lattice* [5]) is illustrated in Fig. 1 for $n = 4$.

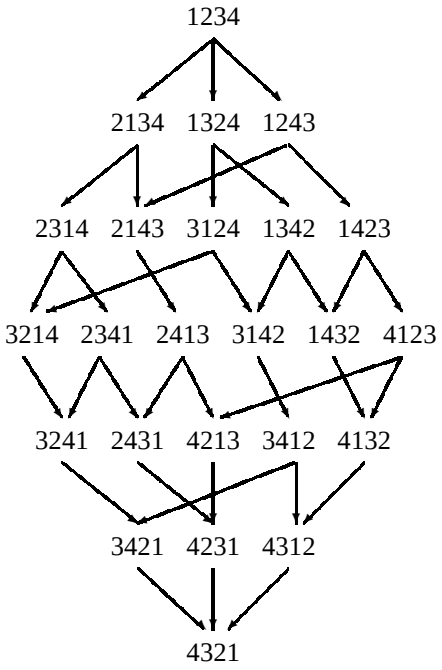

Figure 1. Lattice of permutations for $n = 4$

Notice that this graph, which contains $n!$ nodes, is not implemented in the proposed method.

In terms of scheduling problems, each node of this graph corresponds to a sequence and a successor of sequence σ is a sequence σ' where two consecutive jobs have been interchanged. In the following we will talk about 'sequence' instead of 'permutation'.

2.2 Basic properties

This graph structure has already been studied in the literature and the properties that are indicated below are well known [3, 11, 6].

The graph is a succession of $n(n-1)/2 + 1$ levels. We say that sequence $(n, n-1, ..., 1)$ is at level 0 and that sequence $(1, 2, ..., n)$ is at level $n(n-1)/2$, denoted by:

$$\kappa(n, n-1, ..., 1) = 0,\ \kappa(1, 2, ..., n) = \frac{n(n-1)}{2}$$

We denote by

$$\Gamma(\sigma) = \{(i, j)/i < j \text{ and } i \prec_\sigma j\}$$

Property 1 [3]: For any sequence σ at level κ, the level of sequence σ is exactly the number of its inversions (transposition of two neighbors), i.e. the number of times we have $i \prec_\sigma j$ with $i < j$:

$$\kappa(\sigma) = |\Gamma(\sigma)|$$

Illustration: For instance, let consider sequence $\sigma = (1, 2, 3, 4)$. We have $1 \prec_\sigma 2$ and $1<2$, $1 \prec_\sigma 3$ and $1<3$, $1 \prec_\sigma 4$ and $1<4$, $2 \prec_\sigma 3$ and $2<3$, $2 \prec_\sigma 4$ and $2<4$, $3 \prec_\sigma 4$ and $3<4$, and therefore, the level of sequence $(1, 2, 3, 4)$ is 6. For sequence $(4, 1, 2, 3)$ we only have $1 \prec_\sigma 2$ and $1<2$, $1 \prec_\sigma 3$ and $1<3$, $2 \prec_\sigma 3$ and $2<3$ and therefore the level of sequence $(4, 1, 2, 3)$ is equal to 3.

Property 2 [3]: Let consider a sequence σ at level κ and the set $\Gamma(\sigma)$. Any predecessor π of σ in the graph is such that:

$$\Gamma(\sigma) \subset \Gamma(\pi)$$

Illustration: Let consider sequence $(4, 1, 2, 3)$. We have $\Gamma(4, 1, 2, 3) = \{(1, 2), (1, 3), (2, 3)\}$. We have $\Gamma(1, 4, 2, 3) = \{(1, 4), (1, 2), (1, 3), (2, 3)\}$.

Property 3: Let consider a sequence σ at level κ and the set $\Gamma(\sigma)$. $\Gamma(\sigma)$ gives the characteristics of all the predecessors of σ and no other sequence satisfies these characteristics.

Illustration: Let consider sequence $(4, 1, 2, 3)$ and $\Gamma(4, 1, 2, 3) = \{(1, 2), (1, 3), (2, 3)\}$. All the predecessors of $(4, 1, 2, 3)$ are such that $1 \prec 2$ and $1 \prec 3$ and $2 \prec 3$ (i.e. 1, 2, 3 are in this order and 4 can be put at any position). All the other sequences in S_4 do not satisfy this set of constraints.

3. Method

The method giving the characteristics of the set of optimal sequences for some given problems is based on the properties of the lattice of permutations. We suppose that $(1, 2, ..., n)$ is an optimal sequence and that a rule exists for defining the relative position of two consecutive jobs. Such rules exist actually for some scheduling problems and 'pairwise interchange' arguments are used for proving the optimality of these sequences (see Section 4).

If we are able to find an optimal sequence σ_1 at a level as small as possible in the graph, then by using a trivial pairwise interchange argument, we can prove that all its predecessors in the graph are also optimal sequences. And there is no need to give the list of these predecessors, since we know that they are characterized by $\Gamma(\sigma_1)$.

In order to characterize the complete set of optimal sequences, we have to search another sequence σ_2 with minimal level, that is not characterized by σ_1. And we iterate the search with σ_3, etc., until no further sequence can be found.

The general method works as follows:

1. Solve the scheduling problem with the fitted rule.

2. Renumber the jobs so that $(1, 2, ..., n)$ is the optimal sequence
3. Repeat
   (a) Find the optimal sequence σ with minimum level by an algorithm A
   (b) Deduce the characteristics of the predecessors of σ
   (c) Prevent these sequences from being obtained again by A
4. Until no sequence can be found by A
5. Return the characteristics of all the sequences obtained

Two new scheduling problems can be identified with this approach:

- PB1: assuming that $(1, 2, ..., n)$ is an optimal sequence, find an optimal sequence with minimum level.
- PB2: assuming that $(1, 2, ..., n)$ is an optimal sequence, find an optimal sequence with minimum level which fulfills a set of constraints (find a sequence which is not already characterized).

The scheduling problems that we consider can be solved in polynomial time by a trivial sorting algorithm. However, the complexity of these new problems remains open. The algorithm A which is used in this study is an MIP solver.

4. Application to scheduling problems

This method can be used for solving several scheduling problems, for which a sequencing rule exists for deciding the order of two consecutive jobs. The notations are the following. We consider a set of n jobs to schedule, we denote by p_j, r_j and d_j the processing time, the release date and the due date of job j, respectively, $1 \leq j \leq n$. C_j denotes the completion time of job j. The criteria that will be used in the rest of the paper are the makespan $C_{\max} = \max_{1 \leq j \leq n} C_j$, the maximum lateness $L_{\max} = \max_{1 \leq j \leq n}(C_j - d_j)$, and the total [weighted] completion time $\sum[w_j]C_j = \sum_{1 \leq j \leq n}[w_j]C_j$.

The method works for some scheduling problems like the $1||L_{\max}$, the $1|r_j|C_{\max}$, the $1||\sum[w_j]C_j$, the $F2||C_{\max}$, the $F2|S_{nsd}|C_{\max}$ (where S_{nsd} stands for no-sequence dependent setup times), etc. For all these problems, a sequencing rule allows finding an optimal solution in polynomial time ($O(n \log n)$).

In this paper, the method is developed for a single machine problem denoted by $1||L_{\max}$ and for the classical two-machine flowshop problem denoted by $F2||C_{\max}$. In the $1||L_{\max}$ problem, each job has a due date and the problem is to find a sequence so that the maximum lateness of jobs is minimum. The problem is solved by the rule *earliest due date first* or Jackson's rule [9], i.e. by sorting the jobs in the non-decreasing order of d_j. In the $F2||C_{\max}$ problem, each job is composed by two operations. The first operation is processed on the first machine denoted by M_1 and then the second operation is processed on the second machine denoted by M_2. The problem is solved to optimality by the Johnson's rule [10].

4.1 General model – First sequence

We present in this section two MIP models which allow us to find the first sequence.

4.1.1 MIP1: positional variables

We define $x_{j,k}$ as a Boolean variable equal to 1 if job j is in position k, 0 otherwise. We denote by $n_{j,k}$ a continuous variable equal to the number of constraints obtained if j is in position k in the sequence.

The objective is to minimize the level of the sequence (see Property 1):

$$\text{MIN} \sum_{j=1}^{n} \sum_{k=1}^{n} n_{j,k} \tag{1}$$

There is one job per position and one position per job:

$$\sum_{j=1}^{n} x_{j,k} = 1, \forall k \in \{1, 2, ..., n\} \tag{2}$$

$$\sum_{k=1}^{n} x_{j,k} = 1, \forall j \in \{1, 2, ..., n\} \tag{3}$$

Fix the variables $n_{j,k}$:

$$n_{j,k} \geq \sum_{j'=j+1}^{n} \sum_{k'=k+1}^{n} x_{j',k'} - HV(1 - x_{j,k}),$$
$$\forall j \in \{1, 2, ..., n-1\}, \forall k \in \{1, 2, ..., n-1\} \tag{4}$$

If j is in position k, $x_{j,k} = 1$ and $n_{j,k}$ is equal to the number of times, a job with an index greater than j is sequenced after j (it corresponds to amount κ previously defined). HV is a high value, which can be set for our problem to $n(n-1)/2$.

4.1.2 MIP2: relative position variables

We define $y_{i,j}$ as a Boolean variable equal to 0 if job i precedes job j and 1 otherwise. We also introduce continuous variables: t_j is the start time of job j for the $1||L_{\max}$ problem, and $t_{j,1}$ and $t_{j,2}$ are the start times of job j on machine M_1 and on machine M_2, respectively, for the $F2||C_{\max}$ problem. In order to minimize the level of the sequence, the objective to maximize is:

$$\text{MAX} \sum_{i=1}^{n} \sum_{j=i+1}^{n} y_{i,j} \tag{5}$$

In addition, it is assumed that the Boolean variables verify a kind of triangle inequality:

$$y_{i,k} \leq y_{i,j} + y_{j,k}, \forall i,j,k \in \{1,...,n\}, i \neq j \neq k \quad (6)$$

4.2 Application to $1||L_{\max}$ and $F2||C_{\max}$

The constraint ensuring that the sequence that is returned is optimal, depends on the scheduling problem. $L^*_{\max}$ and $C^*_{\max}$ are the optimal values of the maximum lateness and the makespan, respectively.

4.2.1 MIP1

For the $1||L_{\max}$ problem, we have:

$$\sum_{\ell=1}^{k}\sum_{j=1}^{n} p_j x_{j,\ell} - \sum_{j=1}^{n} d_j x_{j,k} \leq L^*_{\max}, \quad \forall k \in \{1,2,...,n\} \quad (7)$$

For the $F2||C_{\max}$ problem, we denote by $p_{j,1}$ and $p_{j,2}$ the processing time of job j on machine M_1 and on machine M_2, respectively. We have:

$$\sum_{k=1}^{\ell}\sum_{j=1}^{n} p_{j,1} x_{j,k} + \sum_{k=\ell}^{n}\sum_{j=1}^{n} p_{j,2} x_{j,k} \leq C^*_{\max}, \quad \forall \ell \in \{1,2,...,n\} \quad (8)$$

4.2.2 MIP2

For the $1||L_{\max}$ problem, two sets of constraints are needed for the expression of the disjunctive constraints:

$$t_j \geq t_i + p_i - HV y_{i,j}, \quad (9)$$

$$t_i \geq t_j + p_j - HV(1 - y_{i,j}), \quad (10)$$

$$(9) \text{ and } (10), \forall i,j \in \{1,...,n\}, i \neq j$$

Then, we have:

$$t_j + p_j - d_j \leq L^*_{\max}, \forall j \in \{1,...,n\} \quad (11)$$

For the $F2||C_{\max}$ problem, four sets of constraints are needed for the expression of the disjunctive constraints:

$$t_{j,1} \geq t_{i,1} + p_{i,1} - HV y_{i,j}, \quad (12)$$

$$t_{i,1} \geq t_{j,1} + p_{j,1} - HV(1 - y_{i,j}), \quad (13)$$

$$t_{j,2} \geq t_{i,2} + p_{i,2} - HV y_{i,j}, \quad (14)$$

$$t_{i,2} \geq t_{j,2} + p_{j,2} - HV(1 - y_{i,j}), \quad (15)$$

$$(12)...(15), \forall i,j \in \{1,...,n\}, i \neq j$$

Then, we have:

$$t_{j,2} + p_{j,2} \leq C^*_{\max}, \forall j \in \{1,...,n\} \quad (16)$$

4.3 General method – Other sequences

Suppose that the first sequence (σ_1) with minimum level has been obtained. From this sequence, we can deduce the predecessors in the lattice of permutations: the sequences satisfying a set of precedence constraints of type ($a_1 \prec b_1$) and ($a_2 \prec b_2$) and ... and ($a_\nu \prec b_\nu$).

In order to obtain another sequence with minimum level, which is not yet characterized, we introduce the following constraints: ($b_1 \prec a_1$) or ($b_2 \prec a_2$) or ... or ($b_\nu \prec a_\nu$).

4.3.1 MIP1

The expression of these constraints is the following:

$$\sum_{k=1}^{n} k \times x_{b_1,k} \leq \sum_{k=1}^{n} k \times x_{a_1,k} + HV(1 - z_{1,1}) \quad (17)$$

$$\sum_{k=1}^{n} k \times x_{b_2,k} \leq \sum_{k=1}^{n} k \times x_{a_2,k} + HV(1 - z_{1,2}) \quad (18)$$

$$\vdots$$

$$\sum_{k=1}^{n} k \times x_{b_\nu,k} \leq \sum_{k=1}^{n} k \times x_{a_\nu,k} + HV(1 - z_{1,\nu}) \quad (19)$$

$$\sum_{h=1}^{\nu} z_{1,h} \geq 1 \quad (20)$$

with $z_{1,h}$ a Boolean variable equal to 1 if condition number h holds, $1 \leq h \leq \nu$. Constraint (17) ensures that the position of job b_1 is smaller than the position of job a_1, i.e. $b_1 \prec a_1$. Constraint (20) imposes that at least one condition is satisfied, and thus that sequence σ_1 and its predecessors cannot be obtained.

This iteration introduces ν new Boolean variables and $\nu + 1$ constraints. The new sequence obtained is denoted by σ_2; the predecessors of this sequence are characterized, new Boolean variables $z_{2,h}$ and new constraints are introduced, and the process iterates until no more sequence can be found.

4.3.2 MIP2

The expression is the following:

$$y_{a_1,b_1} + y_{a_2,b_2} + ... + y_{a_\nu,b_\nu} \geq 1 \quad (21)$$

Each iteration only produces one additional constraint and does not generate any additional variable.

4.4 Generalization to ρ-approximated sequences

The procedure presented before allows us to characterize the set of optimal sequences. It is possible to use this method to characterize sub-optimal sequences, with a guaranteed performance.

For a given objective performance f, we define a ρ-approximated sequence ($\rho \in \mathbb{R}^+$) as a sequence σ such that:

$$f^* \leq f(\sigma) \leq (1 + \rho) \times f^*$$

if $f^* > 0$, with f^* the criterion value of the optimal sequence ($\rho = 0$ imposes that σ is optimal), and

$$f^* \leq f(\sigma) \leq (1-\rho) \times f^*$$

if $f^* < 0$.

It is sufficient to replace $L^*_{\max}$ in (7) and $C^*_{\max}$ in (8) by $(1+\rho) \times L^*_{\max}$ (or $(1-\rho) \times L^*_{\max}$ if $L^*_{\max} < 0$) and $(1+\rho) \times C^*_{\max}$, respectively.

5. Computational experiments

The size of model MIP1 for problems $1||L_{\max}$ and $F2||C_{\max}$ are the same: n^2 Boolean variables, n^2 continuous variables, and $3n + (n-1)^2$ constraints for the first iteration. At iteration k, the size is: $n^2 + \sum_{h=1}^{k} \nu_h$ Boolean variables, n^2 continuous variables, and $3n + (n-1)^2 + \sum_{h=1}^{k}(\nu_h + 1)$ constraints, with ν_h in $O(n^2)$.

With model MIP2, we need n^2 Boolean variables for both problems and n or $2n$ continuous variables (for the $1||L_{\max}$ or the $F2||C_{\max}$, respectively), at any iteration. The number of constraints is equal to $n^3 + 2n(n-1) + n + (k-1)$ and $n^3 + 4n(n-1) + 2n + (k-1)$ for each problem at iteration k, respectively.

Computational experiments were carried out to evaluate our propositions. Processing times have been randomly generated in the interval $[1, 100]$ and the due date of job J_j have been generated in $[p_j, 1.2 \times \sum p_i]$. The tests were performed with CPLEX 12.1 on a PC clocked at 2.26 GHz with 3.45 GB RAM. 30 instances have been generated per value of n. Four values have been considered for ρ: $\rho \in \{0, 5\%, 10\%, 20\%\}$.

The computation time has been limited to 600 seconds, for finding the whole set of sequences for each instance. The average values only concern the instances that have been solved within the timeout.

The results are reported in Table 1 for the single machine problem and in Table 2 for the two-machine flowshop problem. The last columns ('unslv') indicate the number of unsolved instances. First of all, we remark that model MIP2 clearly outperforms model MIP1. This is due to the size of the model, which increases quickly for model MIP1 with the number of sequences found. However, even for the best MIP model, it is clear that CPLEX is already out for small-scale problems. This is due to the problems difficulty, which increases with the number of constraints that are inserted in the model, and to the important number of sequences that are needed to characterize the total set of ρ-approximated sequences.

In Table 3, the average and maximum number of sequences needed to characterize the whole set of sequences is reported (only for the instances solved within the time limit). We can see that for small size problems, the number of sequences needed to characterize all the ρ-approximated sequences can be already important (more than 360 for 8 jobs in the case of the two-machine flowshop problem and more than 130 for 10 jobs in the case of

$1\|\|L_{\max}$		cpu (s)		unslv	
n	ρ (%)	MIP1	MIP2	MIP1	MIP2
6	0	0.36	0.18	0	0
	5	0.37	0.16	0	0
	10	0.45	0.18	0	0
	20	0.57	0.26	0	0
8	0	2.55	0.69	0	0
	5	2.70	0.81	0	0
	10	10.34	0.78	0	0
	20	18.09	0.80	0	0
10	0	60.25	26.33	11	1
	5	64.82	37.69	10	1
	10	115.95	20.14	14	2
	20	84.09	25.89	17	2

Table 1. Computation times for the single machine problem

$F2\|\|C_{\max}$		cpu (s)		unslv	
n	ρ (%)	MIP1	MIP2	MIP1	MIP2
6	0	0.30	0.75	0	0
	5	1.36	0.87	0	0
	10	2.24	1.18	0	0
	20	3.85	1.23	0	0
8	0	111.65	27.95	3	0
	5	212.11	84.76	20	0
	10	235.52	108.09	25	1
	20	50.89	59.78	18	0

Table 2. Computation times for the two-machine flowshop problem

the single machine problem). This clearly limits the use of CPLEX for solving this very difficult problem considering larger instances.

		$1\|\|L_{\max}$		$F2\|\|C_{\max}$	
n	$\rho\,(\%)$	avg	max	avg	max
6	0	2.30	7	3.40	11
	5	2.37	7	7.67	20
	10	2.47	7	10.50	23
	20	2.90	11	12.27	32
8	0	4.97	16	28.50	96
	5	5.27	16	79.97	224
	10	6.37	19	109.83	227
	20	6.67	21	78.97	362
10	0	22.24	104	–	–
	5	22.59	132	–	–
	10	22.36	85	–	–
	20	25.68	85	–	–

Table 3. Average and maximum number of sequences

It can be also noticed that, for a given number of jobs, the results are not linked to the value of ρ, since the CPU time and the number of unsolved problems remain stable.

It is also interesting to notice that there is no straightforward relation between the evolution of the number of sequences, needed to characterize the whole set of ρ-approximated solutions, and the value of ρ.

6. Conclusion

We have considered in this paper an original discrete optimization problem, which consists in characterizing the whole set of ρ-approximated sequences for some scheduling problems. The problem of characterizing solutions has not receive a great interest in the scheduling literature, despite its theoretical and practical interests. The problems that can be treated by the proposed method suppose that a sequencing rule exists for deciding the best order of two consecutive jobs. The properties of the lattice of permutations are used for the deduction of the characteristics of the sequences. We search for the ρ-approximated sequence with minimum degree in the lattice (maximum depth) and we iterate until no sequence can be found. The problems complexity remains open.

Two linear programming models are proposed and tested. In the first one, positional Boolean variables are used whereas in the second one, sequencing Boolean variables are used. The computational experiments show that some methods like dedicated branch-and-bound algorithms may certainly have some chances to perform better than MIP solvers.

A future research direction is to go further in the complexity study of the problems and to succeed to prove that they are NP-hard (what we guess). It then be worth proposing new exact solving methods. To that aim, an exact method including constraint programming techniques dedicated to the SATISFIABILITY problem would certainly be very promising for this problem.

Acknowledgements

This work was supported by the ANR project no. 08-BLAN-0331 named “ROBOCOOP”.

References

[1] C. Artigues, J.-C. Billaut, and C. Esswein. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2):314–328, 2005.

[2] C. Artigues, F. Roubellat, and J.-C. Billaut. Characterization of a set of schedules in a resource-constrained multi-project scheduling problem with multiple modes. *International Journal of Industrial Engineering: Theory Applications and Practice*, 6(2):112–122, 1999.

[3] M. Bennett and G. Birkhoff. Two families of Newman lattices. *Algebra Universalis*, 32(1):115–144, 1994.

[4] J.-C. Billaut and F. Roubellat. A new method for workshop real time scheduling. *International Journal of Production Research*, 34(6):1555–1579, 1996.

[5] V. Bowman. Permutation polyhedra. *SIAM J. on Applied Mathematics*, 22(4):580–589, 1972.

[6] V. Duquenne and A. Cherfouh. On permutation lattices. *Mathematical Social Sciences*, 27(1):73–89, 1994.

[7] J. Erschler, F. Roubellat, and J.-P. Vernhes. Finding some essential characteristics of the feasible solutions for a scheduling problem. *Operations Research*, 24(4):774–783, 1976.

[8] P. Esquirol, M.-J. Huguet, and P. Lopez. Modelling and managing disjunctions in scheduling problems. *Journal of Intelligent Manufacturing*, 6(2):133–144, 1995.

[9] J. R. Jackson. Scheduling a production line to minimize maximum tardiness. *Management science research*, 43, 1955.

[10] S. M. Johnson. Optimal two- and three-stage production with setup times included. *Naval Research Quarterly*, 1:61–68, 1954.

[11] G. Markowsky. Permutation lattices revised. *Mathematical Social Sciences*, 27(1):59–72, 1994.

[12] I. Sabuncuoglu and M. Bayiz. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126(3):567–586, 2000.

[13] I. Sabuncuoglu and O. Kizilisik. Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 41(17):4211–4231, 2003.

[14] S. Van de Vonder, F. Ballestin, E. Demeulemeester, and W. Herroelen. Heuristic procedures for reactive project scheduling. *Computers and Industrial Engineering*, 52(1):11–28, 2007.