Robust Machine Scheduling Based on Group of Permutable Jobs

Christian Artigues, Jean-Charles Billaut, Azeddine Cheref, Nasser Mebarki, Zakaria Yahouni

Robust Machine Scheduling Based on Group of Permutable Jobs

Christian Artigues, Jean-Charles Billaut, Azzedine Cheref, Nasser Mebarki, and Zakaria Yahouni

Abstract This chapter presents the "group of permutable jobs" structure to represent set of solutions to disjunctive scheduling problems. Traditionally, solutions to disjunctive scheduling problems are represented by assigning sequence of jobs to each machine. The group of permutable jobs structure assigns an ordered partition of jobs to each machine, i.e. a group sequence. The permutation of jobs inside a group must be all feasible with respect to the problem constraints. Such a structure provides more flexibility to the end user and, in particular, allows a better reaction to unexpected events. The chapter considers the robust scheduling framework where uncertainty is modeled via a discrete set of scenarios, each scenario specifying the problem parameters values. The chapter reviews the models and algorithms that have been proposed in the literature for evaluating a group sequence with respect to scheduling objectives for a fixed scenario as well as the recoverable robust optimization methods that have been proposed for generating robust group sequence

C. Artigues (✉) • A. Cheref
CNRS, LAAS, 7 avenue du Colonel Roche, 31400 Toulouse, France

Univ de Toulouse, LAAS, F-31400 Toulouse, France
e-mail: artigues@laas.fr; acheref@laas.fr

J.-C. Billaut
Laboratoire d'Informatique de l'Université de Tours, 64 avenue jean Portalis, 37200 Tours, France
e-mail: jean-charles.billaut@univ-tours.fr

N. Mebarki
LUNAM, Université de Nantes, IRCCyN Institut de Recherche en Communication et Cybernétique de Nantes, UMR CNRS 6597, Nantes, France
e-mail: nasser.mebarki@univ-nantes.fr

Z. Yahouni
LUNAM, IRCCyN/MELT, Manufacturing Engineering Laboratory of Tlemcen,
Tlemcen University, Tlemcen, Algeria
e-mail: zakaria.yahouni@irccyn.ec-nantes.fr

against scenario sets. The methods based on group sequences are compared with standard robust scheduling approaches based on job sequences. Finally, methods for exploiting group sequences in an industrial context are discussed and an experiment for human decision making in a real manufacturing system based on groups is reported.

List of Notations

- $\mathscr{J}$ set of jobs $\{J_j\}_{j=1,\ldots,n}$
- $\mathscr{M}$ set of machines $\{M_k\}_{k=1,\ldots,m}$
- $p_{k,j}$ duration of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$
- p_j duration of job $J_j \in \mathscr{J}$ (one machine case)
- $C_{k,j}$ completion time of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$
- C_j completion time of job $J_j \in \mathscr{J}$ (one machine case or $C_j = \max_{M_k \in \mathscr{M}} C_{k,j}$).
- r_j release date of job $J_j \in \mathscr{J}$
- d_j due date of job $J_j \in \mathscr{J}$
- $L_j = C_j - d_j$ lateness of job $J_j \in \mathscr{J}$
- $C_{\max} = \max_{J_j \in \mathscr{J}} C_j$ makespan
- $L_{\max} = \max_{J_j \in \mathscr{J}} L_j$ maximum lateness
- σ_{ERD} sequence of jobs according to the earliest release date rule
- σ_{EDD} sequence of jobs according to the earliest due date rule
- $\mathscr{S}$ set of uncertainty scenarios
- r_j^s release date of job $J_j \in \mathscr{J}$ under scenario $s \in \mathscr{S}$
- d_j^s due date of job $J_j \in \mathscr{J}$ under scenario $s \in \mathscr{S}$
- p_j^s duration of job $J_j \in \mathscr{J}$ under scenario $s \in \mathscr{S}$ (one machine case)
- $p_{k,j}^s$ duration of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$ in scenario $s \in \mathscr{S}$
- $r_{k,j}^s$ release date of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$ in scenario $s \in \mathscr{S}$
- $d_{k,j}^s$ due date of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$ in scenario $s \in \mathscr{S}$
- $\mathscr{C}^s$ set of feasible schedules for scenario $s \in \mathscr{S}$
- $C_{k,j}$ completion time of job $J_j \in \mathscr{J}$ on machine $M_k \in \mathscr{M}$
- π_j^k index of the machine that precedes machine M_k in job J_j routing
- Ω_j^k index of the machine that follows machine M_k in job J_j routing.
- $L_{\max}^s(C)$ maximum lateness in scenario $s \in \mathscr{S}$ of a schedule $C \in \mathscr{C}^s$
- σ_i^k index of the job at position i on machine $M_k \in \mathscr{M}$ in job sequence σ
- $\mathscr{C}^s(\sigma)$ set of feasible schedules compatible with job sequence σ in scenario s
- $EC^s(\sigma)$ earliest feasible schedule compatible with job sequence σ in scenario s
- $L_{\max}^s(\sigma)$ maximum lateness of $EC^s(\sigma)$ in scenario s
- Σ set of feasible job sequences (scenario independent)
- G group sequence (ordered partition of jobs on each machine)
- ν_k number of groups on machine $M_k \in \mathscr{M}$
- G_i^k i^{th} group on machine $M_k \in \mathscr{M}$
- $\mathscr{G}$ the set of (scenario-independent) feasible group sequences
- $\Sigma(G)$ set of job sequences compatible with group sequence G

- $\mathscr{C}^s(G)$ set of schedules compatible with group sequence G in scenario s
- $\mathscr{EC}^s(G)$ set of earliest schedules compatible with group sequence G in scenario s
- g_j^k position of the group containing $O_{k,j}$
- $\theta_{k,j}$ lower bound for the earliest starting time of $O_{k,j}$ in a group sequence (head)
- $\chi_{k,j}$ lower bound on the earliest completion time of $O_{k,j}$ in a group sequence
- $\theta'_{k,j}$ lower bound on the time between $C_{k,j}$ and $C_{\max}$ (tail)
- $\gamma(G)$ lower bound on the completion time of group G
- $\underline{\tau}_{k,j}$ largest earliest start time of $O_{k,j}$ in a group sequence
- $\underline{C}_{k,j}$ largest earliest completion time of $O_{k,j}$ in a group sequence
- $\overline{\tau}_{k,j}$ smallest latest start time of $O_{k,j}$ in a group sequence
- $\overline{C}_{k,j}$ smallest latest start time of $O_{k,j}$ in a group sequence
- $Prec_j^s$ set of jobs that precede J_j according to the ERD rule in scenario s
- $Succ_j^s$ set of jobs that succeed to J_j according to the ERD rule in scenario s
- $y_{j,q}$ binary variable equal to 1 if job J_j is in the group at position q in the group sequence
- $x_{j,q}$ binary variable equal to 1 if job J_j is at position q in the job sequence
- μ, α, β, ω parameters for instance generation
- $m_{\text{seq}}(O_{k,j})$ free sequential margin of an operation $O_{k,j}$ in a group sequence
- $m_{\text{sn}}(O_{k,j})$ net margin of an operation $O_{k,j}$ in a group sequence
- $m_{\text{sg}}(O_{k,j})$ group margin of an operation $O_{k,j}$ in a group sequence

9.1 Introduction to Scheduling and Robust Scheduling

In this section, disjunctive scheduling problems are defined and notations are introduced. The standard solution representations based on job sequences and disjunctive graphs are reviewed. Robustness definitions are given and the ways robustness can be tackled are also presented.

9.1.1 Scheduling Problems

A scheduling problem consists in defining a set of a start times for a set of tasks that share common resources, taking into account specific time constraints (such as deadlines), and with the aim to optimize an objective function. Sometimes, one also has to decide which resources will perform each task. A schedule is a solution to a scheduling problem (see [8] for a global overview of scheduling problems).

Scheduling problems can be found in all types of organizations or systems. The most famous application domain is the production industry, where scheduling problems take an important place for the production management. Other classical scheduling problems are encountered in computer systems, project management, timetabling, etc. More recent application domains appear, particularly for treating requests in big data environment [4, 5], in hospital environments [14], in rail companies [11], etc.

The theory of scheduling was developed together with the theory of complexity. The methods for solving scheduling problems come from the field of combinatorial optimization, with exact solution methods for finding one optimal solution to the problem and approximate solution methods for finding solutions that are as good as possible.

We denote by $\mathscr{J}$ the set of n jobs to schedule. A job is either a single operation, or is composed by several operations. In this case, a graph allows to define the precedence relations between the operations. In the case of a shop scheduling problem, this graph is called the route or the routing of the job, and it is generally a chain. A set $\mathscr{M}$ of m resources is available for performing the operations. Each resource (called machine in the case of a shop scheduling problem) is supposed to be ready at time 0, and can perform at most one operation at a time. To each operation is associated a performing resource in $\mathscr{M}$ and a processing time, denoted by $p_{k,j}$ for processing time of job J_j on machine M_k (index k is omitted when there is only one resource). It is assumed that the operations are performed without interruption, and that preemption is not allowed.

A schedule is completely characterized by the definition of a vector of starting times, or equivalently by a vector of completion times. We denote by $C_j = \max_{M_k \in \mathscr{M}} C_{k,j}$ the completion time of the last operation of job J_j, $C_{k,j}$ denoting the completion time of job J_j on machine M_k.

The quality of a schedule is given by a performance measure, based on the jobs completion times. The most common objective function is the makespan, denoted by $C_{\max}$ and defined by $C_{\max} = \max_{J_j \in \mathscr{J}} C_j$. If the job J_j is supposed to be finished at date d_j, called the due date of J_j, we define by L_j the lateness of J_j, with $L_j = C_j - d_j$, and $L_{\max} = \max_{J_j \in \mathscr{J}} L_j$, also called the maximum lateness, is another important performance measure for a schedule. Other performance measures can be defined for a schedule, please refer to [8] or [7] for a complete overview of scheduling problems.

In the rest of this chapter, we will consider only two scheduling problems: a scheduling problem with the environment composed by a single machine, and a scheduling problem with a shop environment, where the routes of the jobs are different (also called a *job shop* environment). These two problems are illustrated in the following example.

Example 1: Single Machine Environment

Let consider a single machine environment, and $n = 5$ jobs to schedule. To each job J_j is associated a processing time p_j, a due date d_j, and a release time r_j, which is a date before which a job cannot start. The objective is to schedule these jobs so that the maximum lateness $L_{\max}$ is as small as possible. The data set is given in Table 9.1.

Suppose that the jobs are scheduled in the non decreasing order of the release times. We obtain sequence σ_{ERD}. Suppose that the jobs are scheduled in the non decreasing order of the due dates (also called EDD rule for Earliest Due Date first), we obtain sequence σ_{EDD}. The sequences and their evaluations are given in Fig. 9.1.

Table 9.1: Instance for the single machine problem

J_j	J_1	J_2	J_3	J_4	J_5
r_j	0	7	3	4	3
p_j	3	4	1	2	4
d_j	3	14	4	6	10

Notice that this problem is NP-hard in the strong sense, i.e. no polynomial time algorithm can be proposed for solving the problem to optimality, unless $P = NP$ [13].

Example 2: Job Shop Environment

Let consider now a job shop environment, and $n = 3$ jobs to schedule on $m = 3$ machines and the data given in Table 9.2. To each job J_j are associated exactly 3 operations (one per machine). The objective is to schedule the jobs so that the makespan $C_{\max}$ is as small as possible.

Table 9.2: Instance for the job shop problem

	Machine (duration)		
J_1	$M_1(6)$	$M_2(3)$	$M_3(7)$
J_2	$M_3(8)$	$M_1(6)$	$M_2(4)$
J_3	$M_1(5)$	$M_3(5)$	$M_2(6)$

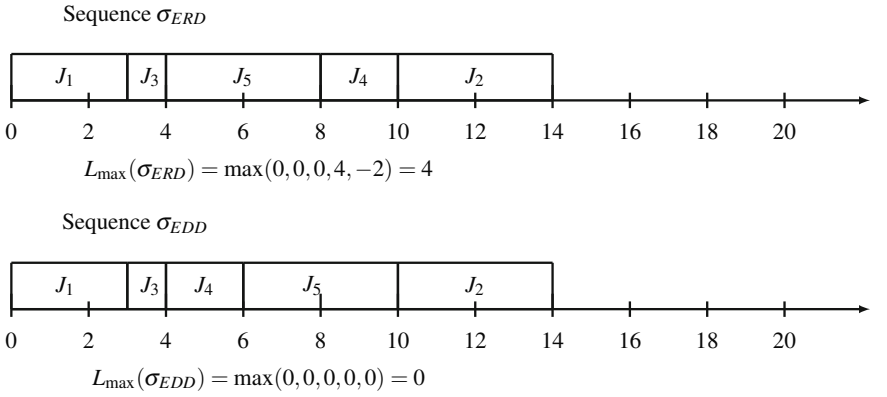


Fig. 9.1: Gantt representation of sequences σ_{ERD} and σ_{EDD} and their evaluations

Figure 9.2 represents a feasible solution of the problem with a makespan equal to 27. Note that the sequences of the jobs on each machine can be different from each other.

9.1.2 Robustness in Scheduling

Robustness considerations receive more and more attention in the literature [6] because in real life situations, unexpected events and uncertainty of the data are challenging the expected plans, making then unusable, sometimes very quickly. This is the reason why a lot of practitioners prefer a robust solution with a lower quality to a vulnerable solution with optimal quality.

Several definitions of the robustness can be found in the literature. We refer to the book by Kouvelis and Yu [15], where a chapter is devoted to robust scheduling problems. As in [15], we consider that there is a significant data uncertainty and the aim is to propose an algorithm returning a solution that hedges against the worst contingency that may arise. Such an approach is called a *robust approach*.

A scenario based approach is used to model the data uncertainty. Each scenario is a data set corresponding to a potential realization. Several scenarios are defined. More formally, let us denote by $\mathscr{S}$ the set of scenarios and s one scenario in $\mathscr{S}$.

We now informally illustrate the concept of robustness on the two scheduling problems that we consider.

Example 1: Single Machine Environment

Let us consider the single machine scheduling problem. We denote by r_j^s, p_j^s and d_j^s the release time, the processing time and the due date of J_j under scenario s. We assume that the previous data set is scenario number 1, and we add a new scenario.

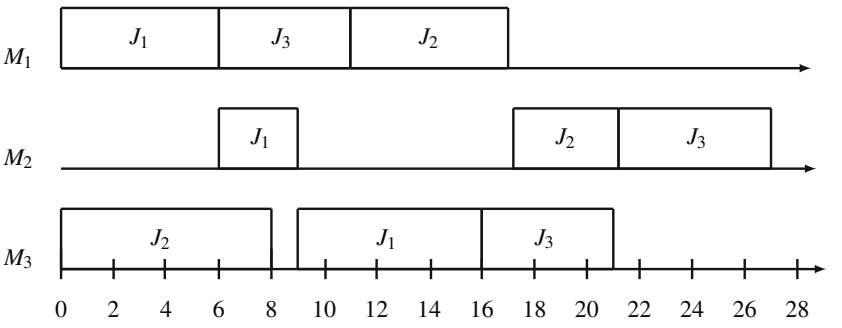


Fig. 9.2: Gantt representation of a solution to the job shop problem

The data set is now given in Table 9.3. Figure 9.1 presents the solutions for the first scenario, we present the sequences σ_{ERD} and σ_{EDD} for the second scenario in Fig. 9.3.

Table 9.3: Instance for the single machine problem with two scenarios

$s=1$	J_1	J_2	J_3	J_4	J_5	$s=2$	J_1	J_2	J_3	J_4	J_5
r_j^1	0	7	3	4	3	r_j^2	3	3	0	1	7
p_j^1	3	4	1	2	4	p_j^2	2	5	1	3	3
d_j^1	3	14	4	6	10	d_j^2	6	11	2	5	14

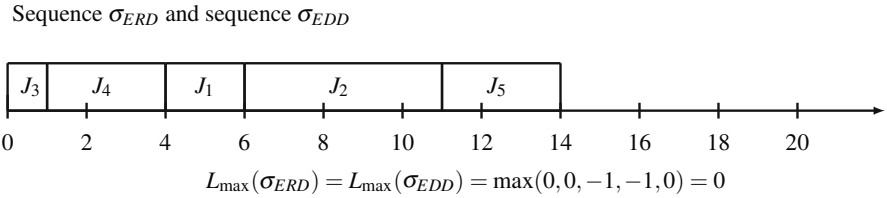


Fig. 9.3: Gantt representation of sequences σ_{ERD} and σ_{EDD} and their evaluations for the second scenario

We can see that for the second scenario, sequences σ_{ERD} and σ_{EDD} are the same: $(J_3, J_4, J_1, J_2, J_5)$. We can say that sequence σ_{EDD} is more robust than sequence σ_{ERD} because the worse evaluation of σ_{EDD} is 0 and the worse evaluation of σ_{ERD} is 4.

Example 2: Job Shop Environment

A second scenario is considered for the job shop problem. The two scenarios are presented in Table 9.4. Only the jobs durations are changed, not the routing of the jobs. Let $p_{k,j}^s$ denote the duration of job J_j on machine M_k in scenario s. If we keep the same sequence on each machine as before and schedule the operations as early as possible, we now obtain the solution represented in Fig. 9.4. The makespan is now equal to 29, which is the worst case makespan on the job sequences.

Table 9.4: Instance for the job shop problem to do

$s=1$	performing machine (duration)			$s=2$	performing machine (duration)		
J_1	$M_1(6)$	$M_2(3)$	$M_3(7)$	J_1	$M_1(8)$	$M_2(2)$	$M_3(5)$
J_2	$M_3(8)$	$M_1(6)$	$M_2(4)$	J_2	$M_3(10)$	$M_1(6)$	$M_2(5)$
J_3	$M_1(5)$	$M_3(5)$	$M_2(6)$	J_3	$M_1(6)$	$M_3(5)$	$M_2(4)$

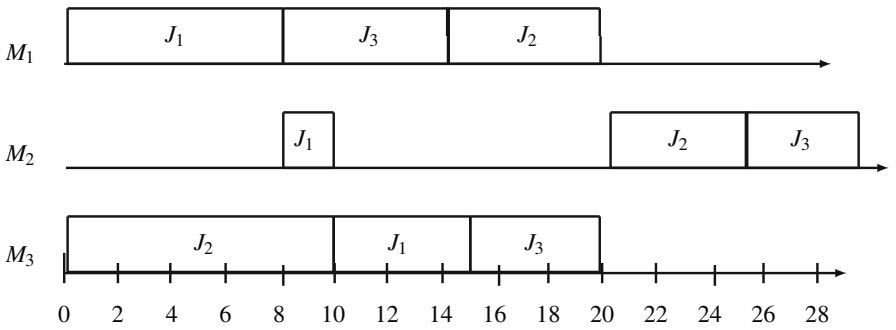


Fig. 9.4: Gantt representation of the same solution to the job shop problem with scenario $s = 2$

9.1.3 Feasible Schedules and the Absolute Robustness Problem

We consider in this section an integration of the job shop scheduling problem and of the single machine problem by defining operation release dates and due dates in the job shop model. More precisely, $r^s_{k,j} \geq 0$ is the release date of the operation of job J_j on machine M_k in scenario s and $d^s_{k,j}$ is the due date of the operation of job J_j on machine M_k in scenario s.

For a scenario $s \in \mathscr{S}$, a feasible solution is a feasible schedule, given by a completion time $C_{k,j}$ of each job J_j on each machine M_k that satisfies:

- Operations release date constraints:

$$C_{k,j} \geq r^s_{k,j} + p^s_{k,j} \quad \forall J_j \in \mathscr{J}, \forall M_k \in \mathscr{M} \tag{9.1}$$

- Jobs routing constraints:

$$C_{k,j} \geq C^s_{\pi^k_j,j} + p^s_{k,j} \quad \forall J_j \in \mathscr{J}, \forall M_k \in \mathscr{M}, \pi^k_j \neq 0 \tag{9.2}$$

  where π^k_j denotes the machine that precedes machine M_k in job J_j routing, with $\pi^k_j = 0$ indicating that machine M_k is the first machine in the routing of job J_j,
- Non-overlapping (also called disjunctive) constraints:

$$C_{k,j} \geq C_{k,i} + p^s_{k,j} \vee C_{k,i} \geq C_{k,j} + p^s_{k,i} \quad \forall J_i, J_j \in \mathscr{J}, i < j, \forall M_k \in \mathscr{M} \tag{9.3}$$

We denote by $\mathscr{C}^s$ the set of feasible schedules for scenario s, i.e. the set of vectors $(C_{k,j})_{M_k \in \mathscr{M}, J_j \in \mathscr{J}}$ that satisfy constraints (9.1)–(9.3). The maximum lateness of a schedule $C \in \mathscr{C}^s$ is given by

$$L^s_{\max}(C) = \max_{J_j \in \mathscr{J}} C_{k,j} - d^s_{k,j}. \tag{9.4}$$

The standard absolute robustness problem (AR) as defined in [15] can now be stated as

$$(AR) \min_{C \in \cap_{s \in \mathscr{S}} \mathscr{C}^s} \max_{s \in \mathscr{S}} L^s_{\max}(C) \tag{9.5}$$

9.1.4 The Standard Solution Representation for (Robust) Disjunctive Scheduling

We note that a schedule may be feasible for a scenario and infeasible for another one. In robust scheduling, it is convenient to consider the concept of job sequence that allows to represent compactly for each scenario a family of feasible schedules. The determination of the feasibility of a job sequence and the computation of the scenario-dependent schedules can be both supported by the classical disjunctive graph representation of the problem [22], defined as follows.

The disjunctive graph has the same vertices and arcs for all scenario but the weights of the arcs are scenario dependent. The disjunctive graph has $nm+2$ vertices, with a vertex jk per operation , for $j = 1,\ldots,n$ and for $k = 1,\ldots,m$ plus dummy start and end vertices 0 and $nm+1$. The disjunctive graph is a 2-graph that contains precedence arcs and disjunctive arcs. For a scenario s, we define a precedence arc $(0, jk)$ between vertex 0 and vertex jk valuated by $r^s_{k,j}$ for each release date constraint (9.1). We define a precedence arc between vertex $j\pi^k_j$ and jk, valuated by $p^s_{k,j}$ for each precedence constraint (9.2). For each disjunctive constraint (9.3), we define two opposite disjunctive arcs, one from node ik to node jk valuated by $p^s_{k,j}$ and one from node jk to node ik valuated by $p^s_{k,i}$. Last we have an arc from each node jk such that $\nexists k', \pi^{k'}_j = k$ and node $nm+1$ valuated by $p^s_{k,j} - d^s_{k,j}$.

A complete selection of the disjunctive graph is, for each pair of disjunctive arcs, the selection of a single arc (and the removal of the opposite one). A complete selection is acyclic if there is no cycle in the graph issued from the selection. Once an acyclic selection is obtained, it defines m total orders of the set of jobs via the selected disjunctive arcs. Hence an acyclic selection can be associated with a job sequence $\sigma = (\sigma^k_i)^{k=1,\ldots,m}_{i=1,\ldots,n}$ where σ^k_i gives the index of the job sequenced at position i on machine M_k. Notice that in the job shop problem a "job sequence" designates in fact a set of job sequences (one per machine). There is a one-to-one mapping between the acyclic selections and the feasible job sequences. Note that in the single machine problem, all job sequences are feasible, which yields to $n!$ job sequences. For the job shop problem, only a subset of job sequences are feasible due to precedence constraints, so we have at most $mn!$ feasible job sequences.

An acyclic complete selection/job sequence σ represents for each scenario a family of feasible schedules $\mathscr{C}^s(\sigma)$ given by the infinite set of potentials in the graph issued from the selection. In this family, a dominant schedule with respect to the $L_{\max}$ objective function is the earliest schedule $EC^s(\sigma)$ such that $EC^s_{k,j}(\sigma)$ is the length of the longest path from vertex 0 to vertex kj in the graph issued from the selection , the corresponding $L^s_{\max}(\sigma)$ is the longest path between 0 and $nm+1$.

Note that given a complete selection, the earliest schedule as well as the cycle determination can be computed in $O(nm)$ by topological sorting and Bellman-Ford algorithm.

Let Σ denote the set of feasible job sequences, which is independent of the scenarios. For a fixed scenario s the standard job shop scheduling problem can be defined as

$$\min_{\sigma \in \Sigma} L^s_{\max}(\sigma)$$

Similarly the absolute robust job shop scheduling problem can be defined as the search for a feasible job sequence that minimizes the worst case maximum lateness

$$(AR) \min_{\sigma \in \Sigma} \max_{s \in \mathscr{S}} L^s_{\max}(\sigma)$$

In terms of robust scheduling, the job sequence σ represents the first stage decision variables that can be taken in advance without knowledge of the realized scenario, while the completion times are the second-stage decision variables that are adjusted according to the realized scenario by picking a schedule in set $\mathscr{C}^s$. In case of an objective function defined as the sum or the maximum of non decreasing job individual functions of the completion time (also called a regular objective function), the earliest schedule $EC^s(\sigma)$ dominates all other schedules of $\mathscr{C}^s$.

Example 1: Single Machine Environment

In Fig. 9.5, we give the disjunctive graph representation of the one-machine problem instance. Because there is no routing constraint between the jobs, the disjunctive arcs form a clique. In Fig. 9.6 we give the conjunctive graph corresponding to the feasible job sequence given by σ_{ERD} for scenario $s = 2$.

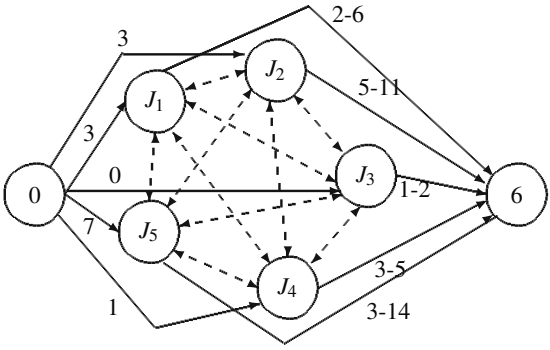
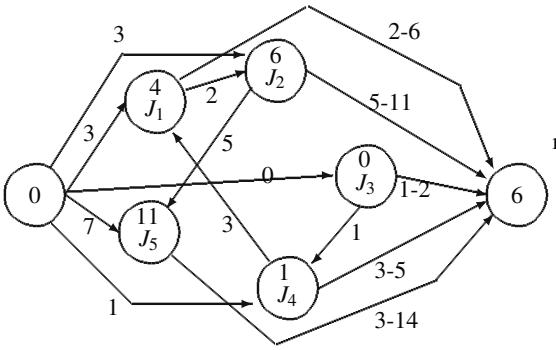


Fig. 9.5: Disjunctive graph of the single machine scheduling problem



max(6-6,11-11,1-2,4-5,14-14)=0

Fig. 9.6: Conjunctive graph corresponding of the solution to the single machine scheduling problem for scenario $s = 2$

Example 2: Job Shop Environment

In Fig. 9.7 we give the disjunctive graph representation of the job shop instance. The conjunctive graph of the complete selection that gives the schedule represented in Fig. 9.2 with scenario $s = 1$ is given in Fig. 9.8.

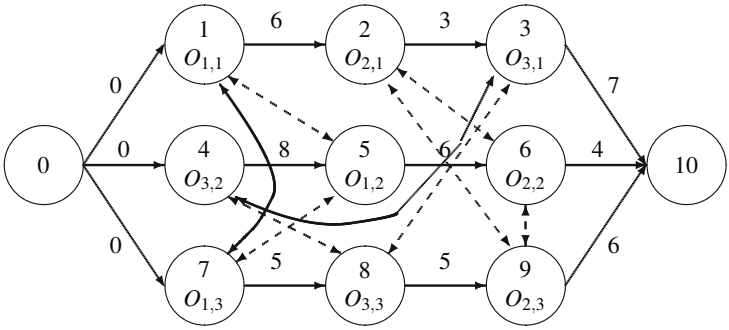


Fig. 9.7: Disjunctive graph corresponding to the instance of the job shop scheduling problem

9.2 Groups of Permutable Jobs: A Solution Structure for Robust Scheduling

This sections present the groups of permutable jobs structure for disjunctive scheduling. Then, it reviews combinatorial optimization problems that have been studied and solved on the group of permutable jobs solution structure.

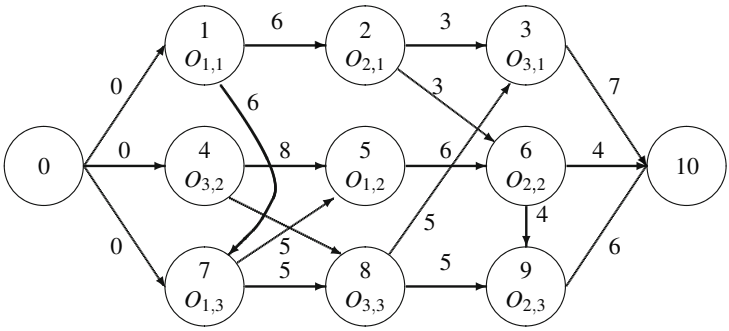


Fig. 9.8: Conjunctive graph corresponding to the solution of the job shop scheduling problem presented for scenario $s = 1$

9.2.1 Groups of Permutable Jobs: A Flexible Solution Representation

A sequence of groups of permutable jobs on a machine M_k is defined as an ordered set partition of the set of jobs $\mathscr{J}$ on machine M_k. An element of each partition is called a group of permutable operations. As for the job sequence, the term "group sequence" is used to name a set of group sequences (one on each machine). This structure was proposed by François Roubellat in the early 1980s [17].

A group sequence $G = (G_i^k)_{i=1,\ldots,\nu_k}^{k=1,\ldots,m}$ with $\cap_{i=1}^{\nu_k} G_i^k = \emptyset$ and $\cup_{i=1}^{\nu_k} G_i^k = \mathscr{J}$ represents a partial job sequence to a disjunctive scheduling problem that specifies on each machine M_k a sequence of ν_k groups of permutable operations, such that G_i^k is the i^{th} group on machine M_k and such that all operations inside a group can be permutated without violating the feasibility of the sequence.

Example 1: Single Machine Environment

Let consider the group sequence presented in Fig. 9.9. This sequence is composed of one group composed by jobs $\{J_1, J_3\}$ and of group composed by jobs $\{J_2, J_4, J_5\}$. Of course, the first group starts at time 3 because J_3 cannot start before date 3. The duration of the group is the sum of the durations of the jobs. One can see easily that whatever the order of the jobs in the first group, the jobs of the second group can be executed at time 7 (or earlier if one starts with job J_1). One can also see that the flexibility provided by this group sequence (12 sequences are characterized) has a price, since the makespan is now equal to 17 instead of 14 before.

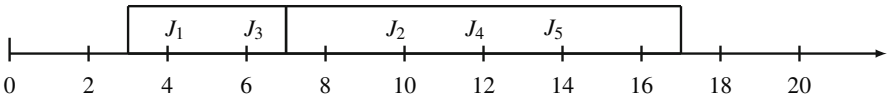


Fig. 9.9: Gantt representation of a group sequence for the single machine problem

Example 2: Job Shop Environment

In Fig. 9.10, two groups of permutable operations are proposed. The first one is composed by the operations of J_1 and J_3 performed on the first machine. The second is composed of the operations of the same jobs on the third machine. One can see that whatever the order of the operations inside each group, the sequence remains feasible. Of course, this flexibility has a price since the makespan is now equal to 32.

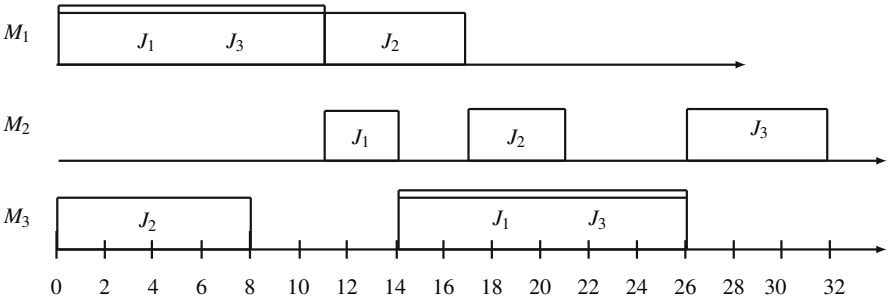


Fig. 9.10: Gantt representation of a group sequence for the job shop scheduling problem

In terms of disjunctive graph, a group sequence matrix corresponds to an incomplete selection that has a particular structure representing the group sequences. Each group is a strongly connected component (via unselected disjunctive arcs) and the selected disjunctive arcs define m totally ordered group sets. We denote by $\mathscr{G}$ the set of (scenario-independent) feasible group sequences.

Given a group sequence $G \in \mathscr{G}$, we denote by $\Sigma(G)$ the set of job sequences that can be obtained from G, and by $\mathscr{C}^s(G)$ the set of schedules issued from the represented job sequences, i.e.:

$$\mathscr{C}^s(G) = \cup_{\sigma \in \Sigma(G)} \mathscr{C}^s(\sigma)$$

For regular objective function, we are interested in the set of earliest schedules that can be issued from a group sequence on a scenario:

$$\mathscr{EC}^s(G) = \cup_{\sigma \in \Sigma(G)} \{EC^s(\sigma)\}$$

It follows that for a group sequence, there is a non unique dominant schedule for a given scenario. Consequently, as a solution representation, a group sequence provides (much) more flexibility. Indeed a group sequence represents $\Pi_{i=1,\ldots,\nu_k}^{k=1,\ldots,m}|G_i^k|$ different job sequences and earliest schedules.

In turn, there is a much bigger number of feasible group sequences. The number of different weak orders on set $\mathscr{J}$ is equal to

$$b_n = \sum_{k=0}^{n} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$

Then, for the single machine problem we have $|\mathscr{G}| = b_n$. For the job shop problem, since not all job sequences are feasible, neither are all group sequences and the number of group sequences is bounded by $|\mathscr{G}| \leq mb_n$.

Finally, we have to remark that the set of feasible job sequences maps the set of feasible group sequences such that each group has a single job.

9.2.2 Combinatorial Optimization Problems on Group Sequences

Several combinatorial optimization problems can be defined on the group sequence solution representation and have been studied in the literature. We restrict to the case where earliest schedules are dominant, which corresponds to regular scheduling objective functions. As a given group sequence G represents in general an exponential number of feasible earliest schedules, given a scenario, a question arises on how to select one schedule among the represented ones. In a robust optimization framework, we assume that, given a disjunctive scheduling problem and a scenario set, a group sequence G is computed as a first-stage decision set. In a second stage decision setting, once the scenario is revealed, we assume that one of the represented job sequence (and its corresponding earliest schedule) is selected with algorithm $A(G,s)$. A typical example would be to define $A(G,s)$ as a list scheduling algorithm that selects an order inside each group according to a priority rule. In fact, we can identify the set of list scheduling algorithms compatible with G and the set of job sequences represented by G. This gives rise to several combinatorial optimization problems.

9.2.2.1 Best Earliest Schedule Within a Fixed Group Sequence for a Fixed Scenario

The objective of the second stage algorithm $A(G,s)$ is naturally to obtain the best schedule according to the realized scenario, which yields problem (GP_1). This gives a lower bound on the performance that an algorithm $A(G,s)$ can reach.

$$(GP_1) \quad \min_{\sigma \in \Sigma(G)} L^s_{\max}(\sigma)$$

As we can define a group sequence G such that $|G| = 1$ for the one machine problem, finding the best schedule in this sequence amounts to solve the one-machine problem itself (denoted by $1|r_i|L_{\max}$), which is an NP-hard problem).

Methods that explicitly solve problem (GP_1) for shop problems in an exact or approximated way can be found in [20, 24]. We illustrate these methods for the makespan objective (by considering that all $d_{k,j}$ are equal to 0). These methods rely on the computation for each operation $O_{k,j}$ of the head of the operation, noted $\theta_{k,j}$, i.e., a lower bound for the earliest starting time, and a tail of the operation, noted $\theta'_{k,j}$, i.e., a lower bound of the time between the operation's latest completion time ($C_{k,j}$) and the end of the schedule ($C_{\max}$).

Heads and tails, which are classical notions in shop scheduling, are adapted for groups of permutable operations. The computation of $\theta_{k,j}$ involves the computation of a lower bound for the earliest completion time for each predecessor of operation $O_{k,j}$: the predecessor operation in job routing ($O_{\pi^k_j,j}$) and the predecessor group on the same machine

Let g^k_j the position of the group containing operation $O_{k,j}$ on machine M_k. Then, $G^k_{g^k_j-1}$ (resp. $G^k_{g^k_j+1}$) is its predecessor (resp. successor) group on machine M_k. $\gamma(G)$ denotes a lower bound of the completion time of group G. The computation of $\gamma(G)$ is based on a one-machine relaxation by making the assumption that each machine has an infinite capacity [20, 24]. Below, a lower bound for the earliest completion time of operation $O_{k,j}$ is denoted $\chi_{k,j}$.

For an operation $O_{k,j}$, its head is computed as follows:

$$\theta_{k,j} = \max(r_{k,j}, \gamma(G^k_{g^k_j-1}), \chi_{\pi^k_j,j}) \tag{9.6}$$
$$\gamma(G^k_i) = C_{\max} \text{ of } 1|r_{k,j}|C_{\max} \text{ with } r_{k,j} = \theta_{k,j}, \quad \forall O_{k,j} \in G^k_i$$
$$\chi_{k,j} = \theta_{k,j} + p_{k,j}$$

Because of the symmetry of heads and tails, tails can be computed as heads using a reversed version of Eq. (9.2.2.1): rather than starting the computation at the beginning of the scheduling problem, the computation begins at the end. We use below symmetrical intermediate values $\gamma'(G)$ and $\chi'_{k,j}$ to compute the tail $\theta'_{k,j}$.

Ω^k_j denotes the machine that follows machine M_k in job J_j routing.

$$\theta'_{k,j} = \max(\gamma'(G^k_{g^k_j+1}), \chi'_{\Omega^k_j,j})$$
$$\gamma'(G^k_i) = C_{\max} \text{ of } 1|r_{k,j}|C_{\max} \text{ with } r_{k,j} = \theta'_{k,j}, \quad \forall O_{k,j} \in G^k_i$$
$$\chi'_{k,j} = \theta'_{k,j} + p_{k,j}$$

These heads and tails can be directly used for the computation of a valid lower bound for any regular objective. For example, for the makespan objective, a lower bound is :

$$\max_{k\in\mathscr{M}} \max_{i=1,\ldots,v_k} (\gamma(G_i^k))$$

For the makespan, this lower bound can be improved using the one-machine relaxation proposed by Carlier and Chretienne [10] also based on the computation of heads and tails. In our case the relaxation is made on the groups instead of machines; for each group a lower bound is computed using the exact method of [9]. The maximum value for all groups represents an improved lower bound for the makespan.

In [20], it was shown that the computation of a lower bound for the best earliest schedule has a complexity of $O(n\log n)$ for any regular objective.

9.2.2.2 Worst Earliest Schedule Within a Fixed Group Sequence for a Fixed Scenario

This problem, denoted (GP_2) seeks to determine the worst performance that any algorithm $A(G,s)$ can achieve on a given scenario. This gives an upper bound on the performance of the second stage algorithm. In conjunction with problem (GP_1), we may obtain a lower and an upper bound of the performance of any second stage algorithm compatible with G on a given scenario.

$$(GP_2) \qquad \max_{\sigma\in\Sigma(G)} L^s_{\max}(\sigma)$$

As for the preceding problem, a limit case is to consider a single group of n jobs for the one-machine problem. In this case, we obtain maximization one-machine problems, which are in general easier than their minimization counterpart as shown in [3] that provide polynomial algorithms and complexity proofs for several cases. Furthermore, in [2], it was shown that (GP_2) is polynomial for any disjunctive scheduling problem. The calculation of this worst earliest schedule relies on the computation of the worst earliest starting time and the worst earliest completion time for each operation denoted $\underline{\tau}_{k,j}$ and $\underline{C}_{k,j}$ respectively.

The computation of $\underline{\tau}_{k,j}$ corresponds to executing this operation at its worst position where all its predecessors are placed at their worst latest time. This problem can be formulated as follows :

$$\underline{\tau}_{k,j} = \max(r_{k,j}, \underline{C}_{\pi^k_j,j}, \max_{O_{k,l}\in G^k_{g^k_j-1}} \underline{C}_{k,l}) \tag{9.7}$$

To compute $\underline{C}_{k,j}$, either the worst earliest completion time of operation $O_{k,j}$ does not depend on another operation of the same group, in this case the first term of the following formula is used. Otherwise, in the worst case, the operation ends last in its group and the second term is used.

$$\underline{C}_{k,j} = \max(\underline{\tau}_{k,j} + p_{k,j}, \max_{O_{k,l} \in G^k_{g^k_j}, l \neq j} \underline{\tau}_{k,l} + \sum_{O_{k,l} \in G^k_{g^k_j}} p_{k,l}) \tag{9.8}$$

The computation of $\underline{\tau}_{k,j}$ and $\underline{C}_{k,j}$ can be performed efficiently by longest path computation in a special conjunctive graph, as described in [2]. We do not describe this graph here for sake of conciseness.

Finally, $\max_{J_i \in \mathscr{J}, M_k \in \mathscr{M}} (\underline{C}_{k,j} - d_j)$ represents the worst case evaluation for the groups of permutable operations for the L_{max} objective. More generally, the worst case evaluation can be computed in polynomial time for regular min-max objectives.

This calculation can also be used as an upper bound for min-sum regular objectives.

9.2.2.3 Worst Latest Schedule Within a Fixed Group Sequence for a Fixed Scenario

This problem, denoted $(GP_2^{'})$ is similar to (GP_2) with due dates for the jobs $(d_{k,j})$. It seeks to determine the worst performance that any algorithm $A(G,s)$ can achieve on a given scenario such that there is no late schedule described by the groups of permutable operations.

For this, it needs the computation of the worst latest starting time $\overline{\tau}_{k,j}$ and the worst latest completion time $\overline{C}_{k,j}$ of an operation $O_{k,j}$ which is similar to the computation of $\underline{\tau}_{k,j}$ and $\underline{C}_{k,j}$. A reverse version of Eqs. (9.7) and (9.8) can be used. Rather than starting from the beginning, the computation starts from the end using the due dates of jobs as explained in [19, 23].

$\overline{\tau}^k_j$ and $\overline{C}^k_j$ can be expressed as follows:

$$\begin{aligned} \overline{C}_{k,j} &= \min(d_{k,j}, \overline{\tau}_{\Omega^k_j,j}, \min_{O_{k,l} \in G^k_{g^k_j+1}} \overline{\tau}_{k,l}) \\ \overline{\tau}_{k,j} &= \min(\overline{C}_{k,j} - p_{k,j}, \min_{\forall O_{k,l} \in G^k_{g^k_j}, l \neq j} C_{k,l} - \sum_{\forall O_{k,l} \in G^k_{g^k_j}} p_{k,l}) \end{aligned} \tag{9.9}$$

9.2.2.4 Flexibility Maximization with a Bounded Objective for a Fixed Scenario

Without any assumption on the second stage algorithm $A(G,s)$, a question arises to maximize a flexibility measure of the group sequence G, denoted $flex(G)$, while ensuring an upper bound UB on the objective. Intuitively, this allows to propose a group sequence such that the largest number of represented job sequences (or with an alternative view, the largest number of compatible second stage algorithms) satisfy the upper bound.This yields problem (GP_3):

$$(GP_3) \quad \max_{G \in \mathscr{G}} flex(G) \text{ s.t. } L^s_{\max}(\sigma) \leq UB, \forall \sigma \in \Sigma(G)$$

As an outcome, we obtain a performance guarantee on the worst earliest schedule for the considered scenario but we have no indication of the best achievable objective, which would require to solve (GP_1). As problem (GP_1) is generally NP-hard, an alternative from computing the best schedule represented by a group sequence is to ask that the group sequence represents a fixed sequence σ_0, which yields

$$(GP_4) \quad \max_{G \in \mathscr{G}} flex(G) \text{ s.t. } \sigma^0 \in \Sigma(G), L^s_{\max}(\sigma) \leq UB, \forall \sigma \in \Sigma(G)$$

Provided that σ^0 is selected as the optimal solution for scenario s, then solving (GP_4) gives for a given scenario a group sequence G of maximal flexibility such that any job sequence σ issued from G (or any list scheduling algorithm compatible with G) verifies $L^s_{\max}(\sigma^0) \leq L^s_{\max}(\sigma) \leq UB$.

Several flexibility measures are available from the literature. The most natural one is the number of represented job sequences/earliest schedules

$$flex_1(G) = |\Sigma(G)|$$

In [2], a surrogate flexibility measure is used, as the number of groups. Indeed it holds intuitively that generating less groups yield more flexibility. We define this measure as

$$flex_2(G) = |G|$$

To normalize this criterion, for a disjunctive problem with m machines we can define.[1]

$$flex_3(G) = \frac{mn - |G|}{mn - m}$$

In case of full flexibility, we have m groups and $flex_3 = 100\,\%$. In case of no flexibility we have mn groups and $flex_3 = 0\,\%$.

In [2], an $O(n^3)$ algorithm has been proposed to solve problem (GP_3) with $flex_1$ for the one-machine problem without release dates, and in the case where the due dates are agreeable, i.e. for any two jobs $i, j \in \mathscr{J}$, $p_i \leq p_j \Leftrightarrow d_i \leq d_j$. The same algorithms solves also (GP_4) with $flex_1$ for the same problem, but without the restriction of agreeable due dates. In the same context, (GP_3) is solved with $flex_2$ by a simpler $O(n \log n)$ algorithm. Adding now release dates, (GP_4) is solved with $flex_1$ by an $O(n^7)$ algorithm and with $flex_2$ by an $O(n^4)$ algorithm. Hence $flex_2$ yields generally simpler problems than $flex1$.

By varying UB, different compromise solutions can be found between the flexibility criterion and the represented schedule performance. This was illustrated for a job in [2], where a (GP_4) was heuristically solved with $flex_2$. The represented job sequence σ^0 was set to the optimal job shop solution and UB was set to different values above $L^s_{\max}(\sigma_0)$. In [12], the two-machine flow-shop, open-shop and job shop

[1] This was initially proposed in [12] for $m = 2$.

problems are studied. A problem dual to (GP_3) with $flex_2$ is also considered, in the sense that it consists in minimizing the makespan while the number of groups is bounded.

$$(GP_5) \qquad \min_{G\in\mathscr{G},\sigma\in\Sigma(G)} L^s_{\max}(\sigma) \text{ s. t. } k_1 \leq |G| \leq k_2$$

For the 2-machine flow shop problem (GP_3) and (GP_5) with $flex_2$ are both NP-hard in the strong sense. However, for any integer $1 \leq k \leq n$ the authors propose a heuristic to build a group sequence G such that $|G| = k$ a job sequence σ such that $\frac{C^s_{\max}(\sigma)}{C^s_{\max}(\sigma^*)} \leq \frac{|G|+1}{|G|}$, where σ^* is the job sequence that minimizes the makespan with no restriction on the number of groups. This is a theoretical bound on the makespan increase when the flexibility is increased. They also propose a heuristic for (GP_4) with $flex_2$. Complexity results are also given for (GP_5) and $flex_2$ for the two machine job shop and flow shop problems.

9.2.2.5 Recoverable Robust Optimization for a Fixed List Scheduling Algorithm on a Scenario Set

If we consider now a fixed list scheduling algorithm $A(G,s)$ that outputs a job sequence that is both compatible with group sequence G and feasible for scenario s, we may seek the group sequence that maximizes the robustness of the earliest schedule selected by the list scheduling algorithm according to the realized scenario. We obtain problem (GP_6).

$$(GP_6) \quad \min_{G\in\mathscr{G}} \max_{s\in S} L^s_{\max}(A(G,s))$$

Note that compared with the job sequence representation, the group sequence representation introduces a third decision level. The first decision level builds the group sequence. The second decision level selects the job sequence. The third decision stage selects the schedule. In Sect. 9.3 we present a method to solve (GP_6) for the single machine problem that we compare with the standard robust scheduling method.

9.3 Solution Methods: A Recoverable Robust Approach Based on Groups of Permutable Operations

Using the concept of recoverable robustness (GP_6) proposed in Sect. 9.2, we present in this section a mixed integer linear program (MILP) and a heuristic method for the maximum lateness minimization on the single machine problem. Given a group sequence and according to the realized scenario, the $A(G,s)$ algorithm schedules the jobs inside a group following the Earliest Release Date (ERD) rule. To evaluate the

interest of the recoverable robust approach, a MILP model as well as a tabu search algorithm are proposed for standard robust scheduling method. Experimental tests are performed and comparisons are given.

9.3.1 MILP Formulation

In order to simulate the ERD rule, a list of predecessors and successors denoted $Prec_j^s$ and $Succ_j^s$ are defined for each job J_j and each scenario s as follows:

$$Prec_j^s = \{J_i \in \mathscr{J} / (r_i^s < r_j^s) \text{ or } (r_i^s = r_j^s \text{ and } i < j)\}$$
$$Succ_j^s = \{J_i \in \mathscr{J} / (r_i^s > r_j^s) \text{ or } (r_i^s = r_j^s \text{ and } i > j)\}$$

We define binary variables $y_{j,q}$ equal to 1 if job J_j is in group G_q (i.e. the group at position q), and 0 otherwise. $C_j^s \geq 0$ is the completion time of job J_j in scenario s.

- Assignment constraints: assign each job to exactly one group

$$\sum_{q=1}^{n} y_{j,q} = 1, \quad \forall j \in \{1,\ldots,n\} \tag{9.10}$$

- Non-overlapping constraints: for a given sequence, the completion time of a job J_j is at least equal to the release date of J_i plus the duration of the jobs that are between J_i and J_j in the sequence, including these two jobs. The following two constraints compute the minimum value of the job completion time under each scenario as set by the list scheduling algorithm. The case were J_i and J_j are in the same group is considered by constraints (9.11), whereas constraints (9.12) consider the case where these two jobs are not in the same group.

$$r_i^s + p_i^s + \sum_{l \in (Succ_i^s \cap Prec_j^s)} p_l^s y_{l,q} + p_j^s - M(2 - y_{i,q} - y_{j,q}) \leq C_j^s,$$
$$\forall i,j,q \in \{1,\ldots,n\}, j \in Succ_i^s, \forall s \in \mathscr{S} \tag{9.11}$$

  According to the ERD rule, the expression $\sum_{l \in (Succ_i^s \cap Prec_j^s)} p_l^s y_{l,q}$ computes the total duration of the jobs between J_i and J_j except p_i and p_j.

$$r_i^s + p_i^s + \sum_{l \in Succ_i^s} p_l^s y_{l,q} + \sum_{f=q+1}^{q'-1} \sum_{l=1}^{n} p_l^s y_{l,f} + \sum_{l \in Prec_j^s} p_l^s y_{l,q'} + p_j^s$$
$$-M(2 - y_{i,q} - y_{j,q'}) \leq C_j^s, \quad \forall i,j,q,q' \in \{1,\ldots,n\}, q' > q, \forall s \in \mathscr{S} \tag{9.12}$$

  The total duration between the jobs J_i and J_j is represented by the expression $\sum_{l \in Succ_i^s} p_l^s y_{l,q} + \sum_{f=q+1}^{q'-1} \sum_{l=1}^{n} p_l^s y_{l,f} + \sum_{l \in Prec_j^s} p_l^s y_{l,q'}$. Given a scenario s and according to the ERD rule, expressions $\sum_{l \in Succ_i^s} p_l^s y_{l,q}$ and $\sum_{l \in Prec_j^s} p_l^s y_{l,q'}$ com-

pute the total duration of J_i successors and the total duration of J_j predecessors, respectively. The remaining expression $\sum_{f=q+1}^{q'-1} \sum_{l=1}^{n} p_l^s y_{l,f}$ compute the total duration of groups between those including the two jobs.

- The maximum lateness of a schedule

$$L_{\max} \geq C_j^s - d_j^s, \quad \forall j \in \{1,\ldots,n\}, \forall s \in \mathscr{S} \tag{9.13}$$

- Objective function

$$\min L_{\max}$$

This model contains n^2 binary variables, $n|\mathscr{S}|+1$ continuous variables and $O(n^4|\mathscr{S}|)$ constraints.

9.3.2 Tabu Search Algorithms

The proposed MILP can only be used to solve small problem instances. To overcome the difficulty of solving large instances, we propose a tabu search heuristic to solve larger instances. The metaheuristic works as follows. Starting from an initial solution and defining the neighborhood structure, the procedure selects the best appropriate neighbor solution. The selected solution can be chosen if it is not in the tabu list. Otherwise this solution is rejected and the procedure seeks another neighbor solution. The process is repeated if the global stopping condition is not reached.

- *Step 1.* Initial solution: the initial sequence of groups is obtained by sorting the jobs in their due dates increasing order over the first scenario $s = 1$. From this sequence, n groups are created, by assigning each job to one group.
- *Step 2.* Selection of the best neighbor: starting with the current group sequence solution, each neighbor is evaluated and the best non tabu is kept.
- *Step 3.* Stopping condition: the algorithm stops when a global time limit, fixed to 30 s, is reached.

In the following, the implementation of the encoding, neighborhood structure and tabu list for the proposed algorithm are respectively described.

9.3.2.1 Encoding

Because the group sequence and the composition of groups do not depend on the scenario, a solution of the problem can simply be encoded by a vector of size n. Let v be the used vector for encoding a solution and g_j the index of the group to which job J_j is assigned on the solution sequence. A complete solution is encoded by assigning a value g_j to vector v for each j, $1 \leq j \leq n$.

9.3.2.2 Neighborhood Definition

We denote by $|G|$ the number of groups. Four neighborhood are defined as follows:

- *Groups swap:* let v_j the position to swap, select v_k and swap the values v_j and v_k. This neighborhood swaps the groups assigned to jobs J_j and J_k.
- *Group insert:* insert a job J_j of group v_j to an existing group v_k.
- *Group split:* split a group into two groups. The fact that the sequence inside the group depends on the realised scenario makes the split infeasible. To rectify it, we order the jobs inside a group in increasing order according to the average due dates of the jobs over the scenarios.
- *Groups fusion:* merge two consecutive groups into one.

9.3.2.3 Tabu List

The tabu list contains solutions that are recently selected and prevents to choose them again. The experiment analysis do not really show the contribution of a tabu list. Therefore, we have fixed a size of $10n$ for the Tabu list, which gives relatively better results.

9.3.3 Solution Algorithms for the Standard Robust Scheduling Method

To evaluate our algorithms for the recoverable robust approach based on groups of permutable jobs, the algorithms presented above are compared with those of the standard robust scheduling method (without groups of permutable operations). In order to do that, we propose a mixed integer linear program (MILP) and a tabu search heuristic for the standard robust scheduling method. The MILP as well as the tabu search proposed algorithms are briefly presented.

Positional variables $x_{j,k} \in \{0,1\}$ are defined for this model and $L_{\max}$ is a continuous variable to minimize. Variable $x_{j,k}$ takes value 1 if the job J_j is in position k, and 0 otherwise.

- Assignment constraints: assign one job at each position, and one position to each job

$$\sum_{j=1}^{n} x_{j,k} = 1, \quad \forall k \in \{1,\dots,n\}$$

$$\sum_{k=1}^{n} x_{j,k} = 1, \quad \forall j \in \{1,\dots,n\}$$

- Guarantee that the worst earliest schedule $L_{\max}$ is larger than the maximum lateness of each scenario

$$L_{\max} \geq \sum_{j=1}^{n} r_j^s x_{j,k} + \sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q} - \sum_{j=1}^{n} d_j^s x_{j,k'},$$
$$\forall k, k' \in \{1, \ldots, n\}, k \geq k', \forall s \in \mathscr{S}$$

- Objective

$$\min L_{\max}$$

This model contains n^2 binary variables, one continuous variable and $O(n^2|\mathscr{S}|)$ constraints.

To make a fair comparison of the two robust solution paradigms, the proposed tabu search algorithm is based on the same principles as before and starts with the same initial solution. In the following, we describe the implementation of the encoding and the neighborhoods structure, without changing the rest of the algorithm.

A vector v' of size n is used to encode a solution in which v'_j represents the position of job J_j in the schedule. The two used neighborhoods are the following:

- *Position swap:* This neighborhood performs an exchange of the positions of two jobs. Let v_j be the position to swap, select v'_k and swap the values v'_j and v'_k.
- *Position insert:* insert a job at position v'_j to position v'_k. The jobs between v'_j and v'_k will be shifted.

9.3.4 Computational Experiments

The algorithms have been evaluated on randomly generated instances using the following scheme. We first generate (p_j^1, r_j^1, d_j^1) for the scenario $s = 1$ which is called "reference scenario", then for each $s \in \mathscr{S}$, $s \neq 1$, uncertainty was generated from the data (p_j^1, r_j^1, d_j^1). Processing times p_j^1 were uniformly generated in the interval $[1, 50]$ and we denote by $P = \sum_{j=1}^{n} p_j^1$ the total processing times on the reference scenario. Release and due dates were generated in the intervals $[1, \mu P]$ and $\left[\alpha - \frac{\beta}{2}P, \alpha + \frac{\beta}{2}P\right]$, respectively. For modelling the uncertainty on the data, for each scenario $s \in \mathscr{S}$, $s \neq 1$, (p_j^s, r_j^s, d_j^s) are generated uniformly from the "reference scenario" by taking values from the intervals $\left[1 - \omega p_j^1, 1 + \omega p_j^1\right]$, $\left[1 - \omega r_j^1, 1 + \omega r_j^1\right]$ and $\left[1 - \omega d_j^1, 1 + \omega d_j^1\right]$, respectively. Parameters μ, α and β take a fixed value $\{0.5\}$, $\{1\}$ and $\{1\}$, respectively. The last parameter ω takes values from the set $\{0.2, 0.4, 0.6\}$. For each couple (n, s) ten instances are generated in which $n \in \{10, 25, 40, 100\}$ and $\mathscr{S} \in \{2, 5, 10\}$. The experiments have been run for the 360 instances on an Intel i7-4770 CPU 3.40 GHz computer with 8 GB.

We evaluated and compared the performance of the solutions obtained by the MILP models and the tabu search algorithms. We call *RRA* the recoverable robust approach and *SRA* the standard robust approach. Let $z_E(A)$ and $z_H(A)$ denote the objective value of the algorithm $A \in \{RRA, SRA\}$ returned respectively by the exacts methods given by MILPs and the heuristic one given by the tabu search algorithm.

In Table 9.5, we provide the aggregate results for each n. The statistics take account of the average CPU times ttb needed to reach the best solution for both tabu search algorithms. The average gap between the exact solution found and the tabu search solution are given by gap columns. gap takes value $(z_H(SRA) - z_E(SRA))/z_H(SRA)$ in the SRA case and $(z_H(RRA) - z_E(RRA))/z_H(RRA)$ in the RRA one. The last column $\Delta = (z_H(SRA) - z_H(RRA))/z_H(SRA)$ gives the average gap between the two tabu search algorithms.

Table 9.5: Experimental comparisons

	SRA		RRA		
n	$ttb(s)$	gap	$ttb(s)$	gap	Δ
10	0.01	1.7 %	0.01	0 %	24.79 %
25	0.14	2.3 %	0.15	–	17.28 %
40	0.39	–	0.52	–	16.83 %
100	3.21	–	12.58	–	13.71 %

The experiments show the limits of MILP models, especially for the recoverable robust approach. In ttb columns, one can see that the RRA tabu search algorithm has a faster growth than the one for SRA. However, the difference remains not very significant and both algorithms have comparable CPU times for considered instances. As expected finally, column $Delta$ reveals the benefits of the recoverable robust approach to obtain better worst case maximum lateness values. This is due to the fact that the RRA can react to the realized scenario thanks to the group structure.

9.4 Using Groups of Permutable Operations in an Industrial Context

As mentioned in Sect. 9.2.2 robust machine scheduling based on groups of permutable operations is composed of three decisions stages. The second stage decision set consists, once the scenario is revealed, in selecting a job sequence and we assume in this chapter that the earliest schedule is selected from a given job sequence. In an industrial context the selection of a job sequence has to be made in a very short time. In order to fulfill this timing constraint, either heuristics adapted to groups of permutable operations may be used (as the ERD list scheduling algorithm used in the previous section) or the selection may be done by a human operator during the execution of the schedule. This section describes several alternative ways of selecting job sequence from a group sequence on a realized scenario (index s is consequently dropped), keeping industrial requirements in mind.

9.4.1 Heuristics for the Reactive Phase of Groups of Permutable Operations

To solve the job shop scheduling problem one of the most common approaches is the use of heuristics based on priority dispatching rules (PDR), which are rules used to select the next job to process from jobs awaiting service on a resource. The lower bounds presented in Sect. 9.2.2 are used to build PDRs for groups of permutable operations when the objective is to minimize the makespan.

First a PDR based on the operation's tail is proposed. The idea is to give more priority to an operation which exhibits a large tail. As the tails of different operations may be equal, the rule Shortest Processing Time (SPT), which aims at selecting the operation with the shortest imminent processing time, is used to break the ties. This heuristic named SQUTAIL (Square Tail) is formulated as:

$$\min(p_{k,j} - \theta'_{k,j})$$

From the lower bounds another PDR is proposed, with the following behavior:

1. For each operation waiting in the queue, a partial group of operations is generated;
2. The lower bound for the makespan is computed for these partial generated schedules;
3. The operation with the lowest lower bound is then chosen.

To break the ties, this rule, named LB (Lower Bound), is combined with either rule SQUTAIL or the PDR Most Work Remaining (MWR) which selects the operation with the highest remaining processing time. The use of MWR together with LB is named LB+MWR and the use of SQUTAIL together with LB is named LB+SQUTAIL.

Another heuristic, which is not based on the lower bounds, is also proposed. Shifting Bottleneck (SB) heuristic, described in [1], is a very effective heuristic in job shop scheduling for the makespan. In order to adapt the shifting bottleneck for groups of permutable operations, the relaxation in one-machine problem, is used. In our case, the algorithm is not applied to the machines but to the groups. As the number of re-optimizations is higher than with the classical SB, better performances are expected as well as higher computation time. Another benefit of the relaxation applied to the groups is that all the computed schedules are feasible contrarily to the classical SB which may give schedules that are not feasible. This heuristic is named SB.

In [19], these heuristics have been evaluated on a well-known benchmark for job shop scheduling, the Lawrence's instances composed of 40 instances of 8 different sizes [16]. They show that these heuristics are very effective to evaluate the makespan with the following ranking:

- SQUTAIL is the less efficient (in average a deviation of 13.7 % from the optimal) but the fastest (less than 0.13 s in average with a maximum of 0.46 s),

- LB+MWR and LB+SQUTAIL are quite similar in performance (an average of 3.6 % from the optimal) and computing time (an average of 0.94 s with a maximum of 3.8 s),
- SB is the most effective, giving the optimal for 17 instances (in average a deviation of 1.5 % from the optimal and a maximum of 3.2 %). It is also more time-consuming (an average of 3.7 s with a maximum of 10.74 s).

Each heuristic has specific strengths and weaknesses. To give in a very short time a solution, SQUTAIL is a good compromise, SB is a very effective heuristic in regards with the performance, LB+MWR and LB+SQUTAIL are in between.

9.4.2 A Multi-Criteria Decision Support System (DSS) for Groups of Permutable Operations

Another approach to select a job sequence is to let a human operator choose in real-time the next operation to process within a group of permutable operations, according to the operator's knowledge of the context. In order to make his choice, the operator needs criteria adapted to groups of permutable operations. In Sect. 9.2.2 a best earliest schedule and a worst earliest schedule evaluation for any regular objective within a group sequence for a fixed scenario have been presented. In [23] an adaptation of the free margin to the groups of permutable operations is presented. The so-called free sequential margins allows to evaluate during the execution of the schedule, the schedule lateness.

9.4.2.1 Free Sequential Margin

The free sequential margin computes for an operation according to its earliest execution, the maximum tardiness which ensures that all schedules enumerated in the group sequence will present no tardiness.[2] The free sequential margin of an operation $m_{\text{seq}}(O_{k,j})$ has two components:

- the operation's net margin $m_{\text{sn}}(O_{k,j})$, which is related to the operation itself regardless the other operations of the group.
- the operation's group margin $m_{\text{sg}}(O_{k,j})$, which is related to the other operations of the group.

The computation of the proper free sequential margin of $O_{k,j}$ corresponds to the difference between its worst latest starting time $\overline{\tau}_{k,j}$ (9.9) and its worst earliest starting time $\underline{\tau}_{k,j}$ (9.7).

Using the worst case earliest starting time and the worst case latest completion time, the free sequential margin can be expressed as follows :

[2] Positive lateness.

$$\begin{cases} m_{\text{seq}}(O_{k,j}) = \min(m_{\text{sn}}(O_{k,j}), m_{\text{sg}}(O_{k,j})) \\ m_{\text{sn}}(O_{k,j}) = \overline{C}_{k,j} - p_{k,j} - \underline{\tau}_{k,j} \\ m_{\text{sg}}(O_{k,j}) = \min_{\forall O_{k,l} \in G^k_{g^k_j}, l \neq j} \overline{C}_{k,j} - \sum_{\forall O_{k,l} \in G^k_{g^k_j}} p_{k,l} - \underline{\tau}_{k,j} \end{cases}$$

For a given group of permutable operations, several situations may occur:

- All the free sequential margins of the current group are positive or zero, in that case whatever the chosen operation, all possible permutations on the group will give schedules with no tardiness. Nevertheless, choosing the operation with the highest group margin permits to maximize the margins.
- There is at least one operation in the group which presents negative free sequential margin. In that case, there may be sequences in this group which give tardiness, but it is also possible to have sequences with no tardiness:
- If all net margins are positive then there may be sequences on this group with no tardiness. It is recommended to execute the operation with the highest group margin in order to increase the negative margins, trying to make them become positive.
- If there is at least one operation with a negative net margin then all possible permutations on the group will give late schedules.

An industrial manufacturing scheduling software named ORDO have been developed in France, based on the concept of groups of permutable jobs and using the free sequential margin indicator at the shop floor. At the early 2000s ORDO was used in more than 70 make-to-order manufacturing companies. The software is described in [21] and more references can be found in the book [17].

9.4.2.2 Multi-Criteria DSS

The free sequential margin is the only criterion used in an industrial context (ORDO software) to help the operator choose an operation within a group of permutable operations. However, with only one criterion at his disposal, the operator has little choice to make his decision.

An experiment conducted at the University of Nantes has tried to evaluate if a DSS composed of several criteria could be more efficient [18]. This experiment was realized on a real manufacturing system that can be represented by a six machines job shop problem. During the first stage decision set, groups of permutable operations with a fixed scenario were computed. In the second stage, 18 students at the end of their bachelor degree in production management studies have played the role of the operator. Each student was asked to schedule in real time a single workstation (the same for each student), by choosing an operation within a group. The objective given to the students was to minimize the tardiness, measured by the L_{max}.

The students were separated in two equal groups. The first one has only the free sequential margin at his disposal while the second group has five different criteria to make his choice:

- The best earliest schedule evaluation, which gives the best predictable quality of the schedule if the operation is chosen
- The worst earliest schedule evaluation, which gives the worst predictable quality of the schedule if the operation is chosen
- The operation's free sequential margin,
- The operation's sequence in the routing,
- The operation's processing time

For the second group, students have to explicitly query for each criterion, one by one. Thus, the criteria used to help the operator for taking his decision are registered.

The performance of the human-machine system is measured through the quality of the decision process and not through the scheduling performance. Indeed, it would only take one "bad" decision to downgrade the scheduling performance. The quality of the decision making process is evaluated through the proportion of "good solutions" taken by the operator. A solution is considered "good" if it is not dominated by another potential choice considering the L_{max}.

The results, presented in form of Boxplots (Fig. 9.11), show that with a multi-criteria DSS, the proportion of good solutions increases, and this effect is significant. Using the multi-criteria system the mean proportion of good solutions is 0.86 while it is 0.78 using only the free sequential margin. However, Fig. 9.12 shows that the free sequential margin remains the most used criterion, the best and the worst case evaluation are the less used.

This experiment indicates that with a multi-criteria DSS the quality of the decision process is better. Concerning the criteria, the free sequential margin remains the dominating criterion. This is not surprising because the instruction was to minimize the tardiness and this criterion measures the capacity to absorb expected delays. The best and the worst earliest schedules are less used. This can be explained by the fact that they have a great anticipation effect contrarily to the operation's sequence and the operation's processing time. These two criteria give direct information on the operation and thus are better understood by the operator.

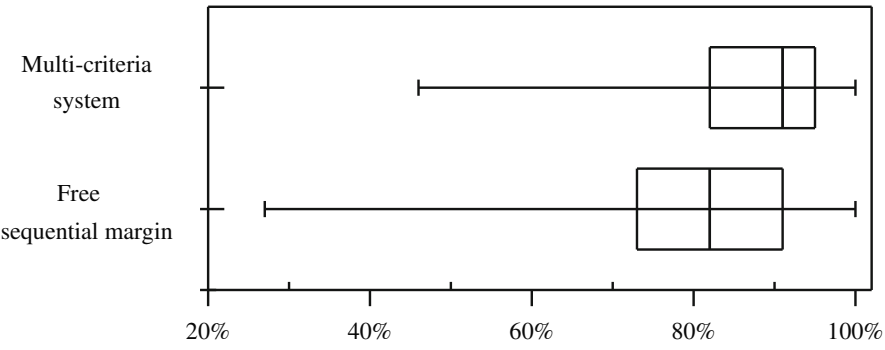


Fig. 9.11: Proportion of good solutions

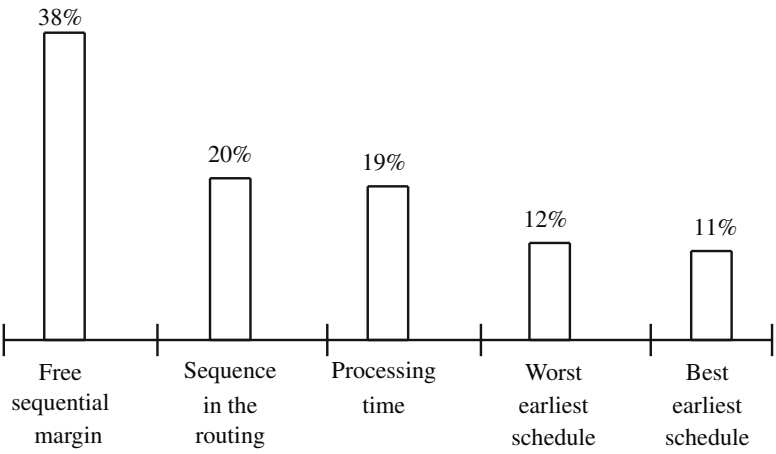


Fig. 9.12: Average proportion of queries by criterion

References

1. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Manag. Sci. **34**(3), 391–401 (1988)
2. Artigues, C., Billaut, J.C., Esswein, C.: Maximization of solution flexibility for robust shop scheduling. Eur. J. Oper. Res. **165**(2), 314–328 (2005)
3. Aloulou, M., Kovalyov, M., Portmann, M.C.: Maximization in single machine scheduling. Ann. Oper. Res. **129**, 21–32 (2004)
4. Berlinska, J., Drozdowski, M.: Scheduling divisible MapReduce computations. J. Parallel Distrib. Comput. **71**(3), 450–459 (2011)
5. Berlinska, J., Drozdowski, M.: Scheduling multilayer divisible computations. RAIRO Oper. Res. **49**(2), 339–368 (2015)
6. Billaut, J.-C., Moukrim, A., Sanlaville, E. (eds.): Flexibility and Robustness in Scheduling. Wiley, New York (2008)
7. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Handbook on Scheduling. From Theory to Applications. Springer, Berlin (2015)
8. Brucker, P.: Scheduling Algorithms, 5th edn. Springer, Berlin (2015)
9. Carlier, J.: The one machine problem. Eur. J. Oper. Res. **11**(1), 42–47 (1982)
10. Carlier, J., Chretienne, P.: An algorithm for solving the job-shop problem. Manag. Sci. **35**(2), 164–176 (1989)
11. D'Ariano A., Pacciarelli, D., Pranzo M.: A branch and bound algorithm for scheduling trains in a railway network. Eur. J. Oper. Res. **183**(2), 643–657 (2007)
12. Esswein, C., Billaut, J.C., Strusevich, V.A.: Two-machine shop scheduling: compromise between flexibility and makespan value. Eur. J. Oper. Res. **167**(3), 796–809 (2005)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co., New York (1979)
14. Kergosien, Y., Lenté, C., Billaut, J.-C., Perrin, S.: Metaheuristic algorithms for solving two interconnected vehicle routing problems in a hospital complex. Comput. Oper. Res. **40**(10), 2508–2518 (2013)
15. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht (1997)
16. Lawrence, S.: Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement). Carnegie-Mellon University, Pittsburgh (1984)

17. Lopez, P., Roubellat, F.: Production Scheduling. ISTE/Wiley, London/Hoboken (2008)
18. Mebarki, N., Cardin, O., Guérin, C.: Evaluation of a new human-machine decision support system for group scheduling. In: 12th IFAC, IFIP, IFORS, IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, vol. 12, no. 1, pp. 211–217 (2013)
19. Pinot, G.: Coopération homme-machine pour l'ordonnancement sous incertitudes. Ph.D. thesis, Nantes University, France (2008)
20. Pinot, G., Mebarki, N.: An exact method for the best case in a group sequence: application to a general shop problem. In: 13th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1269–1274 (2009)
21. Roubellat, F., Billaut, J.-C., Villaumié, M.: Ordonnancement d'atelier en temps réel: d'ORABAID à ORDO. Revue d'automatique et de productique appliquées **8**(5), 683–713 (1995)
22. Roy, B., Sussman, B.: Les problemes d'ordonnancement avec contraintes disjonctives, Note D.S. No. 9 bis, SEMA, Paris (1964)
23. Thomas, V.: Aide à la décision pour l'ordonnancement d'atelier en temps réel. Ph.D. thesis, Paul Sabatier University, France (1980)
24. Yahouni, Z., Mebarki, N., Sari, Z.: New lower bounds for the best-case schedule in groups of permutable operations. IFAC-PapersOnLine **48**(3), 1134–1139 (2015)