Constraint programming for the robust two-machine flow-shop scheduling problem with budgeted uncertainty

Carla Juvin, Laurent Houssin, Pierre Lopez

Constraint programming for the robust two-machine flow-shop scheduling problem with budgeted uncertainty

Carla Juvin[1], Laurent Houssin[1,2], and Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2] ISAE-SUPAERO, Toulouse, France
{carla.juvin, laurent.houssin, pierre.lopez}@laas.fr

Abstract. This paper addresses the robust two-machine permutation flow-shop scheduling problem considering non-deterministic operation processing times associated with an uncertainty budget. The objective is to minimize the makespan of the schedule.
Exact solution methods incorporated within the framework of a two-stage robust optimization are proposed to solve the problem. We first prove that under particular conditions the robust two-machine permutation flow-shop scheduling problem can be solved in polynomial time by the well-known Johnson's algorithm usually dedicated to the deterministic version. Then we tackle the general problem, for which we propose a column and constraint generation algorithm. We compare two versions of the algorithm. In the first version, a mixed-integer linear programming formulation is used for the master problem. In the second version, we use a constraint programming model for the master problem. To the best of our knowledge, the use of constraint programming for a master problem in a two-stage robust optimization problem is innovative.
The experimental results show the very good performance of the method based on the constraint programming formulation. We also notice that Johnson's algorithm is surprisingly efficient for the robust version of the general problem.

Keywords: Flow-Shop Scheduling · Robust Optimization · Uncertainty Budget · Mixed-Integer Linear Programming · Constraint Programming.

1 Introduction

The permutation flow-shop scheduling is a well-studied problem where a set of jobs are to be processed on a set of machines. Each job must be processed on every machine, with given processing times, following the same given order of machines. Each machine can process only one job at a time and the processing sequence of jobs is the same on each machine. This problem is strongly NP-hard for three or more machines. However, the two-machine flow-shop scheduling problem is solved by the well-known Johnson's algorithm [5] for the makespan minimization when operation processing times are assumed deterministic. In

the real world, many sources of uncertainty (processing times variation, machine breakdown, addition of new operations, etc.) can affect the quality and even the feasibility of a schedule.

There exist two major approaches to deal with data uncertainty: stochastic optimization and robust optimization. While stochastic optimization considers probability distribution, robust optimization assumes that uncertain data belong to a given uncertainty set and aims to optimize performance considering the worst-case scenario within that set. The traditional robust optimization approach [8] consists in protecting against the case when all parameters can deviate at the same time, which makes the solution overly conservative. Indeed, there is a very low probability that all parameters take their worst value all together. To overcome this limitation, Bertsimas & Sim introduce an uncertainty budget approach that allows a restriction on the number of deviations that can occur simultaneously to a given budget [2]. In order to reach a trade-off between robustness and solution quality, we exploit this approach to define the uncertainty set. Multi-stage robust optimization has been introduced by Ben-Tal et al. [1]. In some optimization problems, only a part of the decision variables have to be determined before uncertainty is revealed, while the other variables can be chosen after the realization of the uncertainty and can thus be adjusted to the scenario. The authors introduce the *adjustable robust counterpart* where the set of decision variables is split into "here and now" decisions and "wait and see" decisions. Thus, the objective is to find a solution for the "here and now" decision variables such that there always exists "wait and see" variables meeting the constraints for all values of the uncertain parameters, and minimizing the objective value.

The purpose is to find the sequence on the machines (first stage decision), allowing to define a start time for each operation and each scenario (second stage decision), minimizing the makespan in the worst-case scenario considering the budget of uncertainty.

In this paper, we propose exact solution methods to solve the robust two-machine flow-shop scheduling problem. We first study a particular case of the problem. Next, we provide a robust counterpart model based on constraint programming formulation, that is embedded in a column and constraint generation framework. A discussion is conducted on the basis of an analysis of experimental results.

2 Problem Statement

An instance of the two-machine flow-shop scheduling problem implies a set of jobs $\mathcal{J}$ and two machines $\mathcal{M} = \{M_1, M_2\}$. Each job $i \in \mathcal{J}$ consists of two operations $O_{i,1}$ and $O_{i,2}$. The first one, $O_{i,1}$, must be processed by machine M_1 with a duration of $p_{i,1}$ and then, $O_{i,2}$ must be processed by machine M_2 with a duration of $p_{i,2}$. Each machine can process only one job at a time and each job can only be processed on one machine at a time. The objective is to find a permutation of jobs, denoted σ, minimizing the makespan.

When processing times are deterministically known, the problem can be solved in polynomial time by means of Johnson's rule [5], which states that job i must be processed before job j if $\min(p_{i,1}, p_{j,2}) < \min(p_{j,1}, p_{i,2})$.

Here, we consider that the processing times of operations are uncertain. Each processing time $p_{i,m}$ of an operation $O_{i,m},\ i \in \mathcal{J}, m \in \{M_1, M_2\}$, belongs to the interval $[\bar{p}_{i,m}, \bar{p}_{i,m}+\hat{p}_{i,m}]$, where $\bar{p}_{i,m}$ is the nominal value and $\hat{p}_{i,m}$ the maximum deviation of the processing time from its nominal value.

Let Γ be the budget of uncertainty, that is the maximum number of operations whose processing time deviation can occur simultaneously. For each scenario ξ in the set of feasible scenarios $\mathcal{U}^{\Gamma}$, the processing time of operation $O_{i,m}$ is then given by:

$$p_{i,m}(\xi) = \bar{p}_{i,m} + \xi_{i,m} \cdot \hat{p}_{i,m}$$

where $\xi_{i,m}$ is equal to 1 if the processing time of the operation deviates, 0 otherwise.

In this study we consider two types of uncertainty budget:

1. A global budget Γ which denotes the number of operations that can deviate on both machines combined. In this case, the set of feasible scenarios is expressed as:

$$\mathcal{U}^{\Gamma} = \Big\{(\xi_{i,m})_{i\in\mathcal{J},m\in\{M_1,M_2\}} \mid \sum_{i\in\mathcal{J}}\sum_{m=1}^{2} \xi_{i,m} \leq \Gamma,\ \xi_{i,m} \in \{0,1\}\Big\}$$

2. A machine-dependent budget $\Gamma = (\Gamma_1, \Gamma_2)$ where Γ_1 and Γ_2 denote the number of operations whose processing time deviation can occur simultaneously on machines M_1 and M_2, respectively. In this case, the set of feasible scenarios is expressed as:

$$\mathcal{U}^{\Gamma} = \Big\{(\xi_{i,m})_{i\in\mathcal{J},m\in\{M_1,M_2\}} \mid \sum_{i\in\mathcal{J}} \xi_{i,1} \leq \Gamma_1, \sum_{i\in\mathcal{J}} \xi_{i,2} \leq \Gamma_2,\ \xi_{i,m} \in \{0,1\}\Big\}$$

2.1 Worst-case evaluation

For a given sequence of jobs σ, there exists a worst-case scenario, maximizing the value of the makespan. Levorato et al. [6] developed a polynomial-time ($\mathcal{O}(n^2)$) worst-case determination procedure, using dynamic programming, for machine-dependent budget $\Gamma = (\Gamma_1, \Gamma_2)$. The same idea is now used when considering a global budget Γ.

Given a machine $m \in \mathcal{M}$, and two integer numbers $i \in [1, |\mathcal{J}|]$ and $\gamma \in [0, \Gamma]$, let $C^{\gamma}_{i,m}(\sigma)$ be the maximum completion time of the i^{th} job of sequence σ, on machine m, considering at most γ deviations. This value is defined by the following recurrence relations:

$$C^{\gamma}_{i,1}(\sigma) = \max(C^{\gamma}_{i-1,1}(\sigma) + \bar{p}_{\sigma[i],1},\ C^{\gamma-1}_{i-1,1}(\sigma) + \bar{p}_{\sigma[i],1} + \hat{p}_{\sigma[i],1}) \qquad (1)$$

$$C^{\gamma}_{i,2}(\sigma) = \max(C^{\gamma}_{i-1,2}(\sigma) + \bar{p}_{\sigma[i],2},\ C^{\gamma}_{i,1}(\sigma) + \bar{p}_{\sigma[i],2},\ C^{\gamma-1}_{i-1,2}(\sigma) + \bar{p}_{\sigma[i],2} + \hat{p}_{\sigma[i],2},\ C^{\gamma-1}_{i,1}(\sigma) + \bar{p}_{\sigma[i],2} + \hat{p}_{\sigma[i],2}) \quad (2)$$

with $C^{\gamma}_{i,m}(\sigma) = -\infty$ if $\gamma < 0$, $C^{\gamma}_{0,m}(\sigma) = 0$ if $\gamma \geq 0$ and $C^{0}_{i,2}(\sigma) = \bar{p}_{\sigma[i],2} + \max(C^{0}_{i-1,2}(\sigma),\ C^{0}_{i,1}(\sigma))$.

The worst-case makespan, under sequence σ, for a global uncertainty budget Γ, is given by $C^{\Gamma}_{|\mathcal{J}|,2}(\sigma)$.

Notations and definitions

$\mathcal{J}$	: Set of jobs
$\mathcal{M}$	: Set of machines
$O_{i,m}$	: m^{th} operation of job i ($i \in \mathcal{J}$) to be executed on machine $m \in \mathcal{M}$
$p_{i,m}$	: processing time of operation $O_{i,m}$ ($i \in \mathcal{J}, m \in \mathcal{M}$)
σ	: sequence of jobs
$\sigma[i]$	: i^{th} job of sequence σ
Γ	: uncertainty budget
Γ_m	: uncertainty budget on machine $m \in \mathcal{M}$
$\mathcal{U}^{\Gamma}$	: uncertainty set for a given uncertainty budget Γ
ξ	: scenario
$C^{\gamma}_{i,m}(\sigma)$	: maximum completion time of the i^{th} job of sequence σ, on machine $m \in \mathcal{M}$, considering at most γ deviations

3 Special Cases

3.1 Global budget and preserved order of processing times

Proposition 1. *If the order of processing times is preserved through deviation, i.e.,* $\forall (i,i') \in \mathcal{J}^2,\ \forall (m,m') \in \mathcal{M}^2,\ \bar{p}_{i,m} < \bar{p}_{i',m'} \Leftrightarrow \bar{p}_{i,m} + \hat{p}_{i,m} < \bar{p}_{i',m'} + \hat{p}_{i',m'}$, *then a schedule following the Johnson's rule is optimal for any global uncertainty budget* Γ.

Proof. Suppose that σ is an optimal sequence for a given global uncertainty budget Γ, with four consecutive jobs, $\sigma[i-1],\ \sigma[i] = j,\ \sigma[i+1] = k$ and $\sigma[i+2]$ meeting one of the following conditions:

(i) $\bar{p}_{j,1} > \bar{p}_{j,2}$ and $\bar{p}_{k,1} < \bar{p}_{k,2}$
(ii) $\bar{p}_{j,1} < \bar{p}_{j,2}$, $\bar{p}_{k,1} < \bar{p}_{k,2}$ and $\bar{p}_{j,1} > \bar{p}_{k,1}$
(iii) $\bar{p}_{j,1} > \bar{p}_{j,2}$, $\bar{p}_{k,1} > \bar{p}_{k,2}$ and $\bar{p}_{j,2} < \bar{p}_{k,2}$

Note σ' the sequence obtained from σ by pairwise interchange jobs j and k. It suffices to show that under any of the three conditions the makespan of the schedule under σ' is not greater than under σ.

The maximal start time of job $\sigma[i+2] = \sigma'[i+2]$, on machine M_1, considering at most γ deviations, is the same under σ and σ', namely, $C^{\gamma}_{i+1,1}(\sigma) = C^{\gamma}_{i+1,1}(\sigma') = \max(a_1, a_2, a_3)$ with:

$a_1 = C^{\gamma}_{i-1,1}(\sigma) + \bar{p}_{j,1} + \bar{p}_{k,1}$

$a_2 = C^{\gamma-1}_{i-1,1}(\sigma) + \bar{p}_{j,1} + \bar{p}_{k,1} + \max(\hat{p}_{j,1}, \hat{p}_{k,1})$ if $\gamma \geq 1$, 0 otherwise
$a_3 = C^{\gamma-1}_{i-1,1}(\sigma) + \bar{p}_{j,1} + \hat{p}_{j,1} + \bar{p}_{k,1} + \hat{p}_{k,1}$ if $\gamma \geq 2$, 0 otherwise

Thus, the rest of the schedule on machine M_1 is not affected by the pairwise interchange. We now study the effect of the change on the schedule on machine M_2, in particular, the date of availability of machine M_2 to process job $\sigma[i+2]$, i.e., $C^{\gamma}_{i+1,2}(\sigma)$ under the original schedule and $C^{\gamma}_{i+1,2}(\sigma')$ after the interchange.

The completion time of job $\sigma[i+1] = k$ under original schedule σ is $C^{\gamma}_{i+1,2}(\sigma) = \max(b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11})$ with:

$b_1 = C^{\gamma}_{i-1,2} + \bar{p}_{j,2} + \bar{p}_{k,2}$
$b_2 = C^{\gamma-1}_{i-1,2} + \bar{p}_{j,2} + \bar{p}_{k,2} + \max(\hat{p}_{j,2}, \hat{p}_{k,2})$ if $\gamma \geq 1$, 0 otherwise
$b_3 = C^{\gamma-2}_{i-1,2} + \bar{p}_{j,2} + \hat{p}_{j,2} + \bar{p}_{k,2} + \hat{p}_{k,2}$ if $\gamma \geq 2$, 0 otherwise
$b_4 = C^{\gamma}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{j,2} + \bar{p}_{k,2}$
$b_5 = C^{\gamma-1}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{j,2} + \bar{p}_{k,2} + \max(\hat{p}_{j,1}, \hat{p}_{j,2}, \hat{p}_{k,2})$ if $\gamma \geq 1$, 0 otherwise
$b_6 = C^{\gamma-2}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{j,2} + \bar{p}_{k,2} + \max(\hat{p}_{j,1} + \hat{p}_{j,2}, \hat{p}_{j,1} + \hat{p}_{k,2}, \hat{p}_{j,2} + \hat{p}_{k,2})$ if $\gamma \geq 2$, 0 otherwise
$b_7 = C^{\gamma-3}_{i-1,1} + \bar{p}_{j,1} + \hat{p}_{j,1} + \bar{p}_{j,2} + \hat{p}_{j,2} + \bar{p}_{k,2} + \hat{p}_{k,2}$ if $\gamma \geq 3$, 0 otherwise
$b_8 = C^{\gamma}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{k,1} + \bar{p}_{k,2}$
$b_9 = C^{\gamma-1}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{k,1} + \bar{p}_{k,2} + \max(\hat{p}_{j,1}, \hat{p}_{k,1}, \hat{p}_{k,2})$ if $\gamma \geq 1$, 0 otherwise
$b_{10} = C^{\gamma-2}_{i-1,1} + \bar{p}_{j,1} + \bar{p}_{k,1} + \bar{p}_{k,2} + \max(\hat{p}_{j,1} + \hat{p}_{k,1}, \hat{p}_{j,1} + \hat{p}_{k,2}, \hat{p}_{k,1} + \hat{p}_{k,2})$ if $\gamma \geq 2$, 0 otherwise
$b_{11} = C^{\gamma-3}_{i-1,1} + \bar{p}_{j,1} + \hat{p}_{j,1} + \bar{p}_{k,1} + \hat{p}_{k,1} + \bar{p}_{k,2} + \hat{p}_{k,2}$ if $\gamma \geq 3$, 0 otherwise

while the completion time of job $\sigma'[i+1] = j$ on machine M_2 under sequence σ' is $C^{\gamma}_{i+1,2}(\sigma') = \max(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11})$, with:

$c_1 = C^{\gamma}_{i-1,2} + \bar{p}_{k,2} + \bar{p}_{j,2}$
$c_2 = C^{\gamma-1}_{i-1,2} + \bar{p}_{k,2} + \bar{p}_{j,2} + \max(\hat{p}_{k,2}, \hat{p}_{j,2})$ si $\gamma \geq 1$, 0 otherwise
$c_3 = C^{\gamma-2}_{i-1,2} + \bar{p}_{k,2} + \hat{p}_{k,2} + \bar{p}_{j,2} + \hat{p}_{j,2}$ if $\gamma \geq 2$, 0 otherwise
$c_4 = C^{\gamma}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{k,2} + \bar{p}_{j,2}$
$c_5 = C^{\gamma-1}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{k,2} + \bar{p}_{j,2} + \max(\hat{p}_{k,1}, \hat{p}_{k,2}, \hat{p}_{j,2})$ si $\gamma \geq 1$, 0 otherwise
$c_6 = C^{\gamma-2}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{k,2} + \bar{p}_{j,2} + \max(\hat{p}_{k,1} + \hat{p}_{k,2}, \hat{p}_{k,1} + \hat{p}_{j,2}, \hat{p}_{k,2} + \hat{p}_{j,2})$ if $\gamma \geq 2$, 0 otherwise
$c_7 = C^{\gamma-3}_{i-1,1} + \bar{p}_{k,1} + \hat{p}_{k,1} + \bar{p}_{k,2} + \hat{p}_{k,2} + \bar{p}_{j,2} + \hat{p}_{j,2}$ if $\gamma \geq 3$, 0 otherwise
$c_8 = C^{\gamma}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{j,1} + \bar{p}_{j,2}$
$c_9 = C^{\gamma-1}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{j,1} + \bar{p}_{j,2} + \max(\hat{p}_{k,1}, \hat{p}_{j,1}, \hat{p}_{j,2})$ if $\gamma \geq 1$, 0 otherwise
$c_{10} = C^{\gamma-2}_{i-1,1} + \bar{p}_{k,1} + \bar{p}_{j,1} + \bar{p}_{j,2} + \max(\hat{p}_{k,1} + \hat{p}_{j,1}, \hat{p}_{k,1} + \hat{p}_{j,2}, \hat{p}_{j,1} + \hat{p}_{j,2})$ if $\gamma \geq 2$, 0 otherwise
$c_{11} = C^{\gamma-3}_{i-1,1} + \bar{p}_{k,1} + \hat{p}_{k,1} + \bar{p}_{j,1} + \hat{p}_{j,1} + \bar{p}_{j,2} + \hat{p}_{j,2}$ if $\gamma \geq 3$, 0 otherwise

It is clear that $b_1 = c_1$, $b_2 = c_2$ and $b_3 = c_3$.
Under condition (i): with $\bar{p}_{j,1} > \bar{p}_{j,2}$ we get $c_4 \leq b_8$, $c_5 \leq b_9$, $c_6 \leq b_{10}$ and $c_7 \leq b_{11}$; with $\bar{p}_{k,1} < \bar{p}_{k,2}$ we get $c_8 \leq b_4$, $c_9 \leq b_5$, $c_{10} \leq b_6$ and $c_{11} \leq b_7$.
Under condition (ii): with $\bar{p}_{k,1} < \bar{p}_{j,1}$ we get $c_4 \leq b_4$, $c_5 \leq b_5$, $c_6 \leq b_6$ and

$c_7 \leq b_7$; with $\bar{p}_{k,1} < \bar{p}_{k,2}$ we get $c_8 \leq b_4$, $c_9 \leq b_5$, $c_{10} \leq b_6$ and $c_{11} \leq b_7$.
Under condition (iii): with $\bar{p}_{j,1} > \bar{p}_{j,2}$ we get $c_4 \leq b_8$, $c_5 \leq b_9$, $c_6 \leq b_{10}$ and $c_7 \leq b_{11}$; with $\bar{p}_{k,2} > \bar{p}_{j,2}$ we get $c_8 \leq b_8$, $c_9 \leq b_9$, $c_{10} \leq b_{10}$ and $c_{11} \leq b_{11}$.

Thus under each condition, $C^{\gamma}_{i+1,2}(\sigma') \leq C^{\gamma}_{i+1,2}(\sigma)$, and therefore the makespan of the schedule under σ' is not greater than under σ. □

Consequently, it is possible to find an optimal sequence for the robust two-machine permutation flow-shop scheduling problem, with global uncertainty budget and preserved order of processing times, in polynomial time.

3.2 Machine-dependent budget $\boldsymbol{\Gamma = (\Gamma_1, \Gamma_2)}$

In general, Johnson's rule does not lead to an optimal robust schedule when considering a machine-dependent uncertainty budget.

Example 1. Consider a robust two-machine flow-shop problem with 3 jobs with machine-dependent uncertainty budget $\Gamma = (1, 2)$. The intervals $[\bar{p}_{i,m}, \bar{p}_{i,m} + \hat{p}_{i,m}]$ of processing times $p_{i,m}$ of operations $O_{i,m}$, $i \in \mathcal{J}, m \in \mathcal{M}$, are given in Table 1.

Table 1. Numerical example of an instance of a two-machine flow-shop problem: preserved order of operation processing times.

	M_1	M_2
J_1	[6,9]	[8,12]
J_2	[10,15]	[4,6]
J_3	[4,6]	[3,5]

Applying Johnson's rule to this instance yields the sequence $\sigma = \{J_1, J_2, J_3\}$. Given the sequence σ, and considering an uncertainty budget $\Gamma = (1, 2)$, the worst case, for this solution, is that the processing time of job J_2 on machine M_1 and jobs J_2 and J_3 on machine M_2 deviate and take their greatest value. Figure 1 depicts the Gantt chart in this case. The objective function value of this solution reaches a makespan equal to 32.

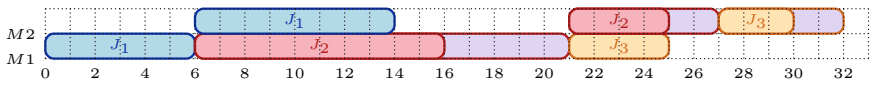

Fig. 1. Example 1 and sequence $\{J_1, J_2, J_3\}$: worst case under Johnson's schedule, $\Gamma_1 = 1, \Gamma_2 = 2$.

Another possible sequence is $\sigma' = \{J_3, J_1, J_2\}$. The worst case for this new solution is such that the processing time of job J_2 on machine M_1 and jobs J_1 and J_2 on machine M_2 deviate from their nominal value. Figure 2 depicts the Gantt chart in this case; it leads to a solution with a worst-case makespan equal to 31.

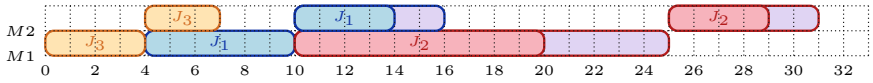

Fig. 2. Example 1 and sequence $\{J_3, J_1, J_2\}$: worst case under optimal robust schedule, $\Gamma_1 = 1, \Gamma_2 = 2$.

Although the order of processing times is preserved, Johnson's rule does not allow to obtain the optimal sequence for this instance when considering a machine-dependent uncertainty budget $\Gamma = (1, 2)$.

3.3 Unpreserved order of processing times

In general, Johnson's rule does not lead to an optimal robust schedule when considering an instance with unpreserved order of processing times, even when we consider a global uncertainty budget.

Example 2. Consider a robust two-machine flow-shop problem with 3 jobs with the global uncertainty budget $\Gamma = 3$. The intervals $[\bar{p}_{i,m}, \bar{p}_{i,m}+\hat{p}_{i,m}]$ of processing times $p_{i,m}$ of operations $O_{i,m}$, $i \in \mathcal{J}, m \in \mathcal{M}$, are given in Table 2.

Table 2. Numerical example of an instance of a two-machine flow-shop problem: unpreserved order of operation processing times.

	M_1	M_2
J_1	[1,5]	[2,3]
J_2	[2,3]	[1,5]
J_3	[2,20]	[4,5]

Applying Johnson's rule to this instance yields the sequence $\sigma = \{J_1, J_3, J_2\}$. Given the sequence σ, and considering an uncertainty budget $\Gamma = 2$, the worst case, for this solution, is such that the processing time of job J_3 on machine M_1 and job J_2 on machine M_2 deviate and take their greatest value. Figure 3 depicts the Gantt chart in this case. The objective function value of this solution reaches a makespan equal to 30.

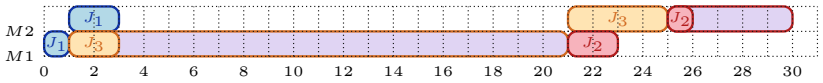

Fig. 3. Example 2 and sequence $\{J_1, J_3, J_2\}$: worst case under Johnson's schedule, $\Gamma = 2$.

Another possible sequence is $\sigma' = \{J_2, J_3, J_1\}$. The worst case for this solution is such that the processing time of job J_2 on machine M_1 and job J_1 on machine M_2 deviate from their nominal value. Figure 4 depicts the Gantt chart in this case; it leads to a solution with a worst-case makespan equal to 29.

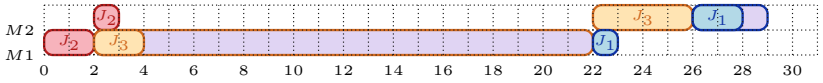

Fig. 4. Example 2 and sequence $\{J_2, J_3, J_1\}$: worst case under optimal robust schedule, $\Gamma = 2$.

Although the considered uncertainty budget is global, Johnson's rule does not allow to obtain the optimal sequence for this instance whose order of processing times is not preserved.

4 General Case

As discussed in the previous section, in the general case, Johnson's rule is not guaranteed to find an optimal robust sequence. However, Mixed-Integer Linear Programming (MILP) or Constraint Programming (CP) allow the development of exact solution methods.

4.1 Mixed-integer linear programming robust counterparts

Levorato et al. [6] proposed two mixed-integer linear programming robust counterparts for the two-machine permutation flow-shop problem.

The first one is adapted from the integer programming model for the three-machine deterministic flow-shop by Wagner [9]. It uses rank decision binary variables, which determine whether a job is placed at a given position in the sequence. It also uses two types of idle times variables. The first ones represent the time each job waits between the end of its execution on machine M_1 and its starting on machine M_2. The others represent the time machine M_2 idles between the execution of each pair of consecutive jobs. These idle times variables

are duplicated for each considered scenario. Precedence constraints are addressed with *job-adjacency* and *machine-linkage* constraints, which exploit the special structure of the problem to describe the relation between idle times, both on machines and jobs, and processing times.

The second robust counterpart proposed by Levorato et al. is based on the formulation presented by Wilson [10]. It also uses rank decision binary variables to determine whether a job is placed at a given position in the sequence. Precedence constraints are based on start time variables defined for each job operation and each machine.

The numerical experiments in [6] highlight the superiority of Wagner's formulation over Wilson's method. Consequently, we only focus on Wagner's formulation and MILP always refers to this formulation in the following.

4.2 Constraint programming robust counterparts

Another alternative is to use constraint programming. To present the CP model, we use the IBM CP Optimizer solver, which allows the use of specific decision variables and constraints. In particular, we use interval and sequence variables defined as follows:

- $task_{i,m,\xi}$: interval variable between the start and the end of the processing of job $i \in \mathcal{J}$ on machine $m \in \{M_1, M_2\}$ in scenario $\xi \in \mathcal{U}^\Gamma$;
- $seqs_{m,\xi}$: sequence variable of operations scheduled on machine $m \in \{M_1, M_2\}$ in scenario $\xi \in \mathcal{U}^\Gamma$.

The CP model developed for the the two-machine robust flow-shop problem is as follows:

$$\min C_{\max} \tag{3}$$

$$\text{s.t.} \quad C_{\max} \geq task_{i,2,\xi}.end \quad \forall i \in \mathcal{J},\ \xi \in \mathcal{U}^\Gamma \tag{4}$$

$$EndBeforeStart(task_{i,1,\xi}, task_{i,2,\xi}) \quad \forall i \in \mathcal{J},\ \xi \in \mathcal{U}^\Gamma \tag{5}$$

$$NoOverlap(seqs_{m,\xi}) \quad \forall m \in \{M_1, M_2\},\ \xi \in \mathcal{U}^\Gamma \tag{6}$$

$$SameSequence(seqs_{1,1}, seqs_{m,\xi}) \quad \forall m \in \{M_1, M_2\},\ \xi \in \mathcal{U}^\Gamma \tag{7}$$

Constraints (4) allow the determination of the makespan, which is equal to the end of the last job on machine M_2 in the worst-case scenario. Constraints (5) ensure the precedence relations between the two operations of a same job. Constraints (6) ensure that, in each scenario, each machine performs at most one operation at a time. Constraints (7) ensure that the sequence is the same on both machines, and the same for each scenario. The first scenario $\xi = 1$ is used as reference, and the constraint is duplicated for each scenario and each machine.

4.3 Column and constraint generation algorithm

The column and constraint generation method has been introduced by Zeng and Zhao [12] to solve two-stage robust optimization problems. The procedure splits the problem into a master problem and an adversarial subproblem. The idea is to solve the robust counterpart problem (or master problem), for a limited subset of scenarios, that fixes the first stage variables, and then to identify which scenarios, if any, make the solution found in the master problem infeasible, using an adversarial subproblem. Then, these scenarios are included in the master problem by generating the corresponding recourse decision variables on the fly. This process repeats until a solution that is feasible for all scenarios is found [3, 4, 6, 7]. Figure 5 depicts the scheme of the column and constraint generation algorithm.

Levorato et al. [6] propose a column and constraint generation framework for the two-machine permutation flow-shop problem. It consists in relaxing one of the MILP formulations presented in Section 4.1 by considering only a subset of scenarios. Then, given a sequence, a polynomial time dynamic algorithm (see Section 2.1) is used to identify the worst-case makespan considering a given uncertainty budget.

Since constraint programming is often very efficient for scheduling problems, we try to improve this framework by replacing the master problem by a relaxed version of the constraint programming model presented in Section 4.2. For this purpose, each robust constraint (4–7) is defined only for a subset of scenarios. The rest of the algorithm remains identical to the version proposed by Levorato et al.

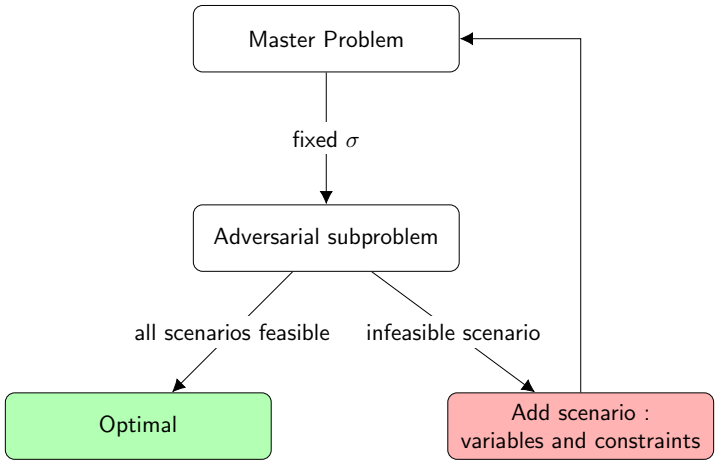

Fig. 5. Column and constraint algorithm.

5 Experimental Results

We evaluate the performance of the column and constraint generation algorithm for both the MILP and the CP models. Experiments are performed on three cluster nodes with Intel Xeon E5-2695 v4 CPU at 2.1 GHz. The algorithms are implemented in C++, CPLEX 12.10 is used as the solver for the MILP master problem and CP Optimizer (CPO) 12.10 for the CP master problem. We limited time to 2 hours, with 4 CPU and a total of 16 GB of RAM, per instance.

5.1 Instances from literature

The instances we used in this section are the same as in [6], based on instances generated by Ying [11]. They are composed of six groups of instances of different size, where the number of jobs $|\mathcal{J}|$ belongs to $\{10, 20, 50, 100, 150, 200\}$. The nominal processing time $\bar{p}_{i,m}$, $i \in \mathcal{J}, m \in \{M_1, M_2\}$ is generated from the uniform distribution $U[10, 50]$ and the processing time deviation $\hat{p}_{i,m}$ is a ratio of the nominal processing time $\alpha\bar{p}_{i,m}$ with $\alpha = 10, 20, 30, 40$ and $50\,\%$. Thus, the order of processing times is preserved through deviation. Ten sets of values for nominal duration were generated for each size $|\mathcal{J}|$, and all deviations ratios α were applied to each of them, giving a total of 300 test instances. The uncertainty budgets, Γ_1 and Γ_2, are set to $20\,\%$, $40\,\%$, $60\,\%$, $80\,\%$ and $100\,\%$ of $|\mathcal{J}|$.

We summarize the results in Table 3 where the performance over instances of different sizes is displayed. We report the percentage of instances solved to optimality before reaching the time limit (*Solved (%)*). For the instances solved to optimality, we display the average execution time, in seconds, to reach optimality (*Avg. time (sec)*). Lastly, we display the average percentage gap of non-optimally solved instances (*Avg. gap (%)*), where the gap is given by:

$$gap^{method} = \frac{UB^{method} - LB^{method}}{UB^{method}}. \tag{8}$$

We notice that the CP model outperforms the MILP one. Indeed, even for the largest instances (200 jobs) the CP-based method manages to solve almost all ($98.83\,\%$) of the instances optimally, while the MILP-based method solves fewer and fewer instances as they grow (down to $68.92\,\%$ for 200 jobs). We also observe that the time needed to reach the optimum is much lower for the CP method, whatever the size of the instances. Finally, we can see that for both methods, the gap is quite low (less than $1\,\%$, regardless of instance size).

Note that the few differences between the results reported here and those presented in [6] concerning the MILP method is probably due to a difference in the implementation of the method and the tools (software and hardware) used for the tests. However, we obtain very comparable results in terms of number of instances optimally solved.

We now examine the quality of the solution obtained by applying Johnson's rule on these instances. Table 4 presents the percentage of best known solution

Table 3. Methods performance comparison grouping by instance sizes.

$\|\mathcal{J}\|$	CP			MILP		
	Solved (%)	Avg. time (sec)	Avg. gap (%)	Solved (%)	Avg. time (sec)	Avg. gap (%)
10	100	0.42	–	100	14.7	–
20	100	0.76	–	100	191.59	–
50	100	1.47	–	98.75	194.9	0.44
100	99.58	0.99	0.09	85.5	319.17	0.64
150	98.92	3.12	0.03	74.67	341.65	0.77
200	98.83	8.49	0.1	68.92	390.73	0.74

found (*Best known sol. (%)*) and the average percentage gap (*Avg. gap (%)*) of non-optimally solved instances ($UB^{method} > LB^*$), where the gap is computed as follows:

$$gap^{method} = \frac{UB^{method} - LB^*}{UB^{method}} \tag{9}$$

and LB^* is the best bound found among the two versions of the column and constraint generation algorithm.

We note that the polynomial time algorithm enables to find high quality solutions. Indeed, for almost all the instances (94.46 %), Johnson's rule provides a robust solution with the same objective value as the best known solution, and the optimality gap is very low.

Table 4. Johnson's rule performance (instances from literature grouping by size).

$\|\mathcal{J}\|$	Best known sol. (%)	Avg. gap (%)
10	92.58	1.13
20	92.67	0.4
50	98.5	0.29
100	97.67	0.04
150	89.67	0.04
200	95.67	0.04

In view of these results, we notice that these instances are easy to solve, due to their particular structure that preserves the order of the processing times. To overcome this, we generated new instances and the results obtained are presented in the following section.

5.2 New instances

In this section, we generated new instances which are also based on the ones from [11]. The nominal processing times $\bar{p}_{i,m}, i \in \mathcal{J}, m \in \{M_1, M_2\}$ remain the same as in the original instances [11]. However, the processing time deviations $\hat{p}_{i,m}$ are randomly generated to avoid the order preserving of processing times. Let $\bar{p}_{max}$ be the maximum nominal processing time for all operations. For each

operation $O_{i,m}, i \in \mathcal{J}, m \in \{M_1, M_2\}$, we randomly generate a value for $\hat{p}_{i,m}$ within a range from 25 % to 80 % of the value of $\bar{p}_{max}$. We generate in total 60 instances that we use for our tests. Again, the uncertainty budgets, Γ_1 and Γ_2, are set to 20 %, 40 %, 60 %, 80 % and 100 % of $|\mathcal{J}|$.

Table 5 presents the same performance indicators as Table 3 for the new generated instances.

By comparing these two tables, it can be seen that the new instances are more difficult to solve than the ones from literature. Indeed, there is a lower proportion of instances solved optimally, a higher average time needed to reach the optimum, as well as a higher average gap for the unsolved instances, for both methods. However, focusing on the information provided by Table 5, it is noticeable that the CP model still outperforms the MILP one, for all observed indicators.

Table 5. Methods performance comparison grouping by instance sizes (new instances).

$\lvert\mathcal{J}\rvert$	CP			MILP		
	Solved (%)	Avg. time (sec)	Avg. gap (%)	Solved (%)	Avg. time (sec)	Avg. gap (%)
10	100	12.3553	–	100	46.7694	–
20	80	167.114	2.16398	75	1002.05	2.67698
50	60	86.8549	2.65432	49	902.686	4.06579
100	56	310.299	2.49723	45	600.667	5.57048
150	48	271.321	2.32075	32	667.885	5.97788
200	54	99.5653	2.68595	30	827.082	6.03535

Tables 6 and 7 detail the percentage of solved instances according to uncertainty budget $\Gamma = (\Gamma_1, \Gamma_2)$ for the CP-based and the MILP-based column and constraint generation method, respectively.

By comparison of these two tables with each other, we notice that, for all combinations of Γ_1 and Γ_2, except one ($\Gamma_1 = 40\,\%$, $\Gamma_2 = 20\,\%$), the CP model outperforms the MILP one. We also see that the problem is more difficult to solve for medium uncertainty budgets (40 % or 60 %), for both methods. This can be explained by the fact that these uncertainty budgets generate a greater number of scenarios. However, the number of scenarios is not the only difficulty factor, as we can see, the methods are more efficient in solving instances with a large uncertainty budget. For example, instances with an uncertainty budget of 80 % are better solved than those with a budget of 20 %, while the number of possible scenarios are the same.

6 Conclusion

In this paper, we investigate the robust two-machine flow-shop scheduling problem where the operation processing times are subject to uncertainty. A two-stage robust optimization is used to deal with this uncertainty, where the first stage is

Table 6. Percentage of solved instances according to uncertainty budget $\Gamma = (\Gamma_1, \Gamma_2)$ for the CP-based column and constraint generation method.

Γ_2 \ Γ_1	20 %	40 %	60 %	80 %	100 %
20 %	28.33	25	41.67	70	100
40 %	35	21.67	26.67	81.67	100
60 %	36.67	28.33	28.33	83.33	100
80 %	60	71.67	75	88.33	100
100 %	100	100	100	100	–

Table 7. Percentage of solved instances according to uncertainty budget $\Gamma = (\Gamma_1, \Gamma_2)$ for the MILP-based column and constraint generation method.

Γ_2 \ Γ_1	20 %	40 %	60 %	80 %	100 %
20 %	26.67	26.67	33.33	58.33	100
40 %	23.33	18.33	23.33	65	100
60 %	23.33	21.67	26.67	65	98.33
80 %	35	35	40	60	100
100 %	76.67	83.33	91.67	98.33	–

devoted to fixing the sequencing decisions whilst the second stage determines the start time of the operations. As a main contribution, we show that under specific conditions the problem can be solved in polynomial time. For the general case, we introduce a constraint programming formulation, which we embed in a column and constraint generation decomposition scheme. This method provides the best results compared to a literature algorithm based on a MILP formulation.

References

1. Ben-Tal, A., Goryashko, A., Guslitzer, E., Nemirovski, A.: Adjustable robust solutions of uncertain linear programs. Mathematical Programming **99**(2), 351–376 (2004)
2. Bertsimas, D., Sim, M.: The price of robustness. Operations Research **52**(1), 35–53 (2004)
3. Duarte, J.L.R., Fan, N., Jin, T.: Multi-process production scheduling with variable renewable integration and demand response. European Journal of Operational Research **281**(1), 186–200 (2020)
4. Hamaz, I., Houssin, L., Cafieri, S.: The cyclic job shop problem with uncertain processing times. In: 16th International Conference on Project Management and Scheduling (PMS 2018). pp. 119–122. Rome, Italy (2018)
5. Johnson, S.M.: Optimal two-and three-stage production schedules with setup times included. Naval Research Logistics Quarterly **1**(1), 61–68 (1954)
6. Levorato, M., Figueiredo, R., Frota, Y.: Exact solutions for the two-machine robust flow shop with budgeted uncertainty. European Journal of Operational Research **300**(1), 46–57 (2022)

7. Silva, M., Poss, M., Maculan, N.: Solution algorithms for minimizing the total tardiness with budgeted processing time uncertainty. European Journal of Operational Research **283**(1), 70–82 (2020)
8. Soyster, A.L.: Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research **21**(5), 1154–1157 (1973)
9. Wagner, H.M.: An integer linear-programming model for machine scheduling. Naval Research Logistics Quarterly **6**(2), 131–140 (1959)
10. Wilson, J.: Alternative formulations of a flow-shop scheduling problem. Journal of the Operational Research Society **40**(4), 395–399 (1989)
11. Ying, K.C.: Scheduling the two-machine flowshop to hedge against processing time uncertainty. Journal of the Operational Research Society **66**(9), 1413–1425 (2015)
12. Zeng, B., Zhao, L.: Solving two-stage robust optimization problems using a column-and-constraint generation method. Operations Research Letters **41**(5), 457–461 (2013)