The robust cyclic job shop problem

Idir Hamaz, Laurent Houssin, Sonia Cafieri

The robust cyclic job shop problem

Idir Hamaz[a], Laurent Houssin[a,b,*], Sonia Cafieri[c]

[a] *LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France*
[b] *ISAE-SUPAERO, Université de Toulouse, Toulouse, France*
[c] *ENAC, Université de Toulouse, F-31055 Toulouse, France*

Abstract

This paper deals with the cyclic job shop problem where the task durations are uncertain and belong to a polyhedral uncertainty set. We formulate the cyclic job shop problem as a two-stage robust optimization model. The cycle time and the execution order of tasks executed on the same machines correspond to the *here-and-now* decisions and have to be decided before the realization of the uncertainty. The starting times of tasks corresponding to the *wait-and-see* decisions are delayed and can be adjusted after the uncertain parameters are known. In the last decades, different solution approaches have been developed for two-stage robust optimization problems. Among them, the use of affine policies, row and row-and-column generation algorithms are the most common. In this paper, we propose a branch-and-bound algorithm to tackle the robust cyclic job shop problem with cycle time minimization. The algorithm uses, at each node of the search tree, a robust version of the Howard's algorithm to derive a lower bound on the optimal cycle time. Moreover, we design a row generation algorithm and a column-and-row generation algorithm and compare it to the branch-and-bound method. Finally, encouraging preliminary results on numerical experiments performed on randomly generated instances are presented.

Keywords: Scheduling, Cyclic job shop problem, Robust optimization

*Corresponding author
Email address: houssin@laas.fr (Laurent Houssin)

1. Introduction

Most models for scheduling problems assume deterministic parameters. In contrast, real world scheduling problems are often subject to many sources of uncertainty. For instance, activity durations can decrease or increase, machines can break down, new activities can be incorporated, *etc.* In this paper, we focus on scheduling problems that are cyclic and where activity durations are affected by uncertainty. Indeed, the best solution for a deterministic problem can quickly become the worst one in the presence of uncertainties.

In this work, we focus on the *Cyclic Job Shop Problem* (CJSP) where processing times are affected by uncertainty. Several studies have been conducted on the CJSP in its deterministic setting. The CJSP with identical jobs is studied in [12] and the author shows that the problem is $\mathcal{NP}$-hard and proposes a Branch-and-Bound algorithm to solve the problem. A more general CJSP is investigated in [10], where the author proposes a mixed linear integer programming formulation and presents a Branch-and-Bound procedure to tackle the problem. A general framework for modeling and solving cyclic scheduling problems is presented in [4] and different models for cyclic versions of CJSP are developed . The flexible CJSP, where the assignment of tasks to machines is a part of the decision, is tackled in [11]. However, a few works consider cyclic scheduling problems under uncertainty. The cyclic hoist scheduling problem with processing time window constraints where the hoist transportation times are uncertain has been investigated by [5]. The authors define a robustness measure for cyclic hoist schedule and present a bi-objective mixed integer linear programming model to optimize both the cycle time and the robustness.

Two general frameworks have been introduced to tackle optimization problems under uncertainty: Stochastic Programming (SP) and Robust Optimization (RO). The main difference between the two approaches is that the Stochastic Programming requires the probability description of the uncertain parameters while RO does not. In this paper, we focus on the RO paradigm. More precisely, we model the robust CJSP as a two-stage RO

problem where the task durations are uncertain. The cycle time and the execution order of tasks on the machines corresponding to the *here-and-now* decisions have to be decided before the realization of the uncertainty, while the starting times of tasks corresponding to the *wait-and see* decisions are delayed and can be adjusted after the uncertain parameters are known. In recent years there has been a growing interest in the two-stage RO and more generally in the multi-stage RO. The two-stage RO is introduced in [2], referred to as *adjustable optimization*, to address the over-conservatism of single stage RO models. Unfortunately, the two-stage RO problems tend to be intractable [2]. In order to deal with this issue, the use of affine policies ([2]) and decomposition algorithms ([13], [14], [1]) have been proposed.

This paper deals with the CJSP where the task durations are uncertain and belong to a polyhedral uncertainty set. The objective is to find a minimum cycle time and an execution order of tasks executed on the same machines such that a schedule exists for each possible scenario in the uncertainty set. To tackle the problem we design a Branch-and-Bound algorithm. More precisely, at each node of the search tree, we solve a robust Basic Cyclic Scheduling Problem (BCSP), which corresponds to the CJSP without resource constraints, using a robust version of the Howard's algorithm to get a lower bound. We also propose a heuristic algorithm to find an initial upper bound on the cycle time. Moreover, the proposed approach is validated through results on numerical experiments performed on randomly generated instances are provided and compared with other approaches such as row generation and row and column generation.

This paper is structured as follows. In Section 2, we present both the Basic Cyclic Scheduling Problem and the Cyclic Job Shop Problem in their deterministic case and introduce the polyhedral uncertainty set considered in this study. Section 3.1 describes a Branch-and-Bound (B&B) procedure to solve the robust CJSP. Numerical experiments performed on randomly generated instances are reported and discussed in Section 4. Finally, some concluding remarks and perspectives are drawn in Section 5.

2. Cyclic scheduling problems and their robust counterpart

In this section, we first introduce the Basic Cyclic Scheduling Problem which corresponds to the CJSP without resource constraints. After presenting the deterministic version of this problem we present the uncertainty set that we consider in this paper and recall recent work on a robust version of BCSP. This problem represents a basis for the Branch-and-Bound method designed for solving the robust CJSP. Next, we present the CJSP in its deterministic case. Finally, we present the uncertainty set that we consider in this paper and we formulate the CJSP with uncertain processing times as a two-stage robust optimization problem.

2.1. Basic Cyclic Scheduling Problem (BCSP) and Robust Basic Cyclic Scheduling Problem (RBCSP)

Let $\mathcal{T} = \{1, ..., n\}$ be a set of n generic operations. Each operation $i \in \mathcal{T}$ has a processing time p_i and must be performed infinitely often. We denote by $< i, k >$ The k^{th} occurrence of the operation i and by $t(i, k)$ the starting time of $< i, k >$.

The operations are linked by a set $\mathcal{P}$ of *precedence constraints* (uniform constraints) given by

$$t(i,k) + p_i \leqslant t(j, k + H_{ij}), \quad \forall (i,j) \in \mathcal{P},\ \forall k \geq 1, \tag{1}$$

where i and j are two generic tasks and H_{ij} is an integer representing the depth of recurrence, usually referred to as *height*.

Furthermore, two successive occurrences of the same task i are not allowed to overlap. This constraint corresponds to the non-reentrance constraint and can be modeled as a uniform constraint with a height $H_{ii} = 1$.

A schedule S is an assignment of starting time $t(i, k)$ for each occurrence $< i, k >$ of tasks $i \in \mathcal{T}$ such that the precedence constraints are met. A schedule S is called *periodic* with cycle time α if it satisfies

$$t(i,k) = t(i,0) + \alpha k, \quad \forall i \in \mathcal{T},\ \forall k \geq 1. \tag{2}$$

For the sake of simplicity, we denote by t_i the starting time of the occurrence $< i, 0 >$. Since the schedule is periodic, a schedule can be completely defined by the vector of the starting times $(t_i)_{i \in \mathcal{T}}$ and the cycle time α.

The objective of the BCSP is to find a schedule that minimizes the cycle time α while satisfying precedence constraints. Note that other objective functions can be considered, such as work-in-process minimization ([10]).

A directed graph $G = (\mathcal{T}, \mathcal{P})$, called *uniform graph*, can be associated with a BCSP such that each node $v \in \mathcal{T}$ (resp. arc $(i, j) \in \mathcal{P}$) corresponds to a generic task (resp. uniform constraint) in the BCSP. Each arc $(i, j) \in \mathcal{P}$ is labeled with two values, a *length* $L_{ij} = p_i$ and a *height* H_{ij}.

We denote by $L(c)$ (resp. $H(c)$) the length (resp. height) of a circuit c in graph G, representing the sum of lengths (resp. heights) of the arcs composing the circuit c. Let us recall the necessary and sufficient condition for the existence of a feasible schedule.

Theorem 1 ([10]). *There exists a feasible schedule if and only if any circuit of G has a positive height.*

A graph that satisfies the condition of Theorem 1 is called consistent. In the following, we assume that the graph G is always consistent. In other words, a feasible schedule always exists.

The minimum cycle time is given by the maximum circuit ratio of the graph G that is defined by

$$\alpha = \max_{c \in \mathcal{C}} \frac{L(c)}{H(c)}$$

where $\mathcal{C}$ is the set of all circuits in G. The circuit c with the maximum circuit ratio is called a *critical circuit*. Thus, the identification of the critical circuit in graph G allows one to compute the minimum cycle time.

Many algorithms for the computation of the cycle time and the critical circuit can be found in the literature. A binary search algorithm with time complexity $\mathcal{O}(nm\left(log(n) + log(\max_{(i,j) \in E}(L_{ij}, H_{ij}))\right))$ has been proposed in[8]. An experimental study about maximum circuit ratio algorithms has been presented in [6]. This study shows that the Howard's algorithm is the most efficient among the tested algorithms.

Once the optimal cycle time α is determined by one of the algorithms cited above, the optimal periodic schedule can obtained by computing the longest path in the graph $G = (\mathcal{T}, \mathcal{P})$ where each arc $(i, j) \in \mathcal{P}$ is weighted by $p_i - \alpha H_{ij}$.

The BCSP can also be solved by using the following linear program:

$$\min \qquad \alpha \tag{3}$$

$$s.t. \quad t_j - t_i + \alpha H_{ij} \geq p_i \quad \forall (i, j) \in \mathcal{P} \tag{4}$$

where t_i represents $t(i, 0)$, i.e., the starting time of the first occurrence of the task i. Note that the precedence constraints (4) are obtained by replacing in (1) the expression of $t(i, k)$ given in (2).

2.1.1. Robust version

A robust version of the BCSP, named $\mathcal{U}^\Gamma$-BCSP, is studied in [9]. The authors consider that the processing time values are uncertain and defined by the *budgeted uncertainty set* introduced in [3]. Each processing time p_i of a task $i \in \mathcal{T}$ belongs to the interval $[\bar{p_i}, \bar{p_i} + \hat{p_i}]$, where $\bar{p_i}$ is the nominal value and $\hat{p_i}$ the deviation of the processing time p_i from its nominal value. A binary variable ξ_i is associated to each operation $i \in \mathcal{T}$. The variable ξ_i is equal to 1 if the processing time of the operation i takes its worst-case value, 0 otherwise. The number of processing time deviations that can occur simultaneously is limited by a parameter Γ called *budget of uncertainty*

For a given budget of uncertainty Γ, that is a positive integer representing the maximum number of tasks allowed to take their worst-case values, the processing time deviations can be modeled through the following uncertainty set:

$$\mathcal{U}^\Gamma = \left\{ (\xi_i)_{i \in \mathcal{T}} \Big| \sum_{i=1}^{\mathcal{T}} \xi_i \leq \Gamma, \, \xi_i \in \{0, 1\} \right\}$$

and the processing time of a task $i \in \mathcal{T}$ is described by

$$p_i(\xi) = \bar{p_i} + \xi_i \hat{p_i}, \quad \forall \xi \in \mathcal{U}^\Gamma.$$

We model the Robust Basic Cyclic scheduling problem as a two-stage

robust optimization problem. The mathematical formulation of the R-BCSP is given below.

$$\min \quad \alpha \tag{5}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq p_i(\xi) \quad \forall (i,j) \in \mathcal{P}, \forall \xi \in \mathcal{U}^{\Gamma} \tag{6}$$

The following theorem characterizes the value of the optimal cycle time for $\mathcal{U}^{\Gamma}$-CJSP:

Theorem 2 ([9]). *The optimal cycle time α of the $\mathcal{U}^{\Gamma}$-CJSP is characterized by*

$$\alpha = \max_{c \in \mathcal{C}} \left\{ \frac{\sum\limits_{(i,j) \in c} \bar{L}_{ij}}{\sum\limits_{(i,j) \in c} H_{ij}} + \max_{\xi : \sum_{i \in \mathcal{T}} \xi_i \leq \Gamma} \left\{ \frac{\sum\limits_{(i,j) \in c} \hat{L}_{ij} \xi_i}{\sum\limits_{(i,j) \in c} H_{ij}} \right\} \right\},$$

where $\bar{L}_{ij} = \bar{p_i}$, $\hat{L}_{ij} = \hat{p_i}$ and $\mathcal{C}$ is the set of all circuits in G.

The problem is proved to be polynomial and three exact algorithms are proposed in [9] to solve the problem. Two of them use a negative circuit detection algorithm as a subroutine and the last one is a Howard's algorithm adaptation. Results of numerical experiments show that the Howard algorithm adaptation yields efficient results.

2.2. Cyclic Job Shop Problem (CJSP)

In the present work, we focus on the cyclic job shop problem (CJSP). Contrary to the BCSP, in the CJSP, the number of machines is lower than the number of tasks. As a result, a sequence of the operations executed on the same machine has to be determined. In [10], the authors prove that the problem is in $\mathcal{NP}$.

Each occurrence of an operation $i \in \mathcal{T} = \{1, ..., n\}$ has to be executed, without preemption, on the machine $M_{(i)} \in \mathcal{M} = \{1, ..., m\}$. Operations are grouped on a set of jobs $\mathcal{J}$, where a job $j \in \mathcal{J}$ represents a sequence of generic operations that must be executed in a given order. To avoid overlapping between the tasks executed on the same machine, for each pair

of operations i and j where $M_{(i)} = M_{(j)}$, the following *disjunctive constraint* holds

$$\forall\, i, j \ s.t.\ M_{(i)} = M_{(j)},\ \forall k, l \in \mathbb{N} : t(i,k) \leq t(j,l) \Rightarrow t(i,k) + p_i \leq t(j,l). \quad (7)$$

To summarize, the CJSP is defined by

- a set $\mathcal{T} = \{1, ..., n\}$ of n generic tasks,
- a set $\mathcal{M} = \{1, ..., m\}$ of m machines,
- a processing time p_i for each task $i \in \mathcal{T}$, that has to be executed on the machine $M_{(i)} \in \mathcal{M}$,
- a set $\mathcal{P}$ of precedence constraints,
- a set $\mathcal{D}$ of disjunctive constraints that occur when two tasks are mapped on the same machine,
- a set $\mathcal{J}$ of jobs corresponding to a sequence of elementary tasks. More precisely, a job J_j defines a sequence $J_j = O_{j,1} \dots O_{j,k}$ of operations that have to be executed in that order.

The CJSP can be represented by a directed graph $G = (\mathcal{T}, \mathcal{P} \cup \mathcal{D})$, called *disjunctive graph*. The sequence of operations that belong to the same job are linked by uniform arcs in $\mathcal{P}$ where the heights are equal to 0. Additionally, for each pair of operations i and j executed on the same machine, a disjunctive pair of arcs (i, j) and (j, i) occurs. These arcs are labeled respectively with $L_{ij} = p_i$ and $H_{ij} = K_{ij}$, and $L_{ji} = p_j$ and $H_{ji} = K_{ji}$ where K_{ij} is an occurrence shift variable to determine that satisfies $K_{ij} + K_{ji} = 1$ (see [10] for further details). Note that the K_{ij} variables are integer variables and not binary variables unlike the non-cyclic job shop problem and the allow an easier formulation of the constraint (7). Two dummy nodes s and e representing respectively the start and the end of the pattern are added to the graph. An additional arc (e,s) with $L_{es} = 0$ and

$H_{es} = WIP$ is considered. The WIP parameter is an integer, called a work-in-process, and represents the number of occurrences of a job concurrently executed in the system.

Finally, the CJSP can be solved by the following MIP:

$$\begin{aligned}
\min \quad & \alpha && && (8a)\\
s.c. \quad & t_j - t_i + \alpha H_{ij} \geq p_i && \forall (i,j) \in \mathcal{P} && (8b)\\
& \left.\begin{array}{l} t_j - t_i + \alpha K_{ij} \geq p_i \\ K_{ij} + K_{ji} = 1 \\ K_{ij} \in \mathbb{Z} \end{array}\right\} && \forall\, (i,j) \in \mathcal{D} && (8c)\\
& t_i \geq 0 && \forall i \in \mathcal{T} && (8d)
\end{aligned}$$

where (8b) are the precedence constraints (the same as (4)) and (8c) are the disjunctive constraints that replace (7).

Job	$\mathcal{J}_1$				$\mathcal{J}_2$			
Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Duration	3	2	4	2	2	1	3	2
Machine	M_1	M_1	M_3	M_1	M_2	M_1	M_3	M_2

Table 1: Instance data for Example 1.

Example 1. *Let us consider an example of a job shop with a set* $\mathcal{T} = \{1,2,3,4,5,6,7,8\}$ *of* 8 *tasks,* 2 *jobs,* 3 *machines and a* $WIP = 2$. *The data are described in Table 1. The job* $\mathcal{J}_1$ *is composed by task* 1, 2, 3 *and* 4 *and the job* $\mathcal{J}_2$ *is composed of tasks* 5, 6, 7 *and* 8. *The disjunctive graph associated to this problem is given in figure 1. The solid black lines depict the uniform constraint and the coloured dashed line indicate the disjunctive constraints (for the purpose of clarity only the disjunctive edges of Machine* M_2 *are labelled). Moreover, the Figure 2 represents a non optimal schedule with* $\alpha = 9$. *However, the optimal cycle time is* $\alpha_{opt} = 8$ *and the associated schedule is shown in Figure 3 but it is necessary to mention that several schedules lead to the same cycle time.*

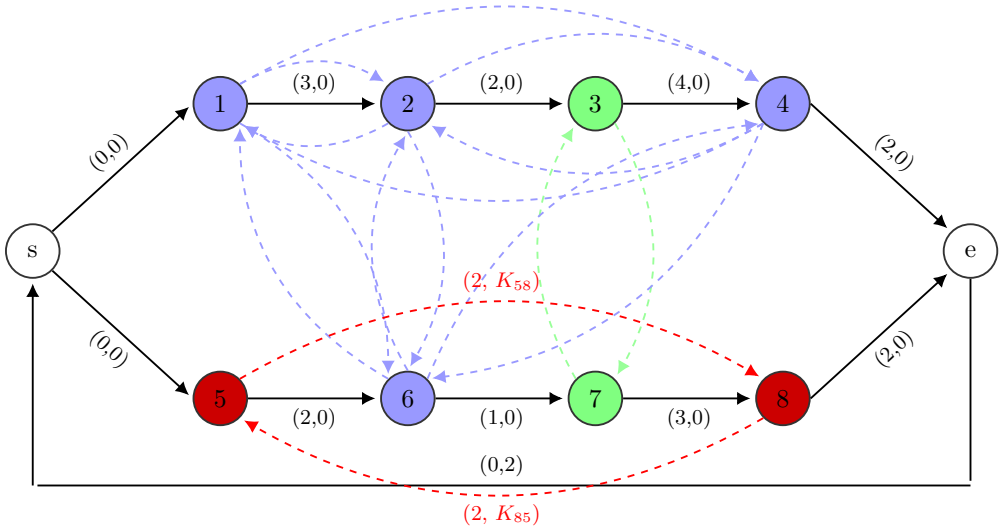


Figure 1: Disjunctive graph associated to instance of Example 1.

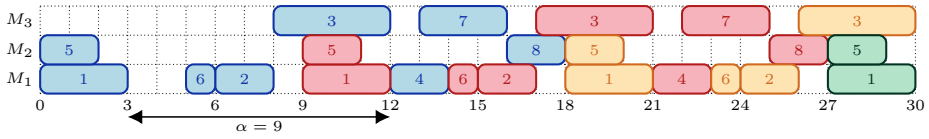


Figure 2: A periodic schedule associated with the CJSP instance with $WIP = 2$.

A lower bound on each occurrence shift value K_{ij} that makes the graph G consistent can be obtained as follows (see [10], [7]):

$$K_{ij}^{-} = 1 - min\{H(\mu) \,|\, \mu \text{ is a path from } j \text{ to } i \text{ in } G = (\mathcal{T}, \mathcal{P} \cup \emptyset)\}. \tag{9}$$

Since $K_{ij} + K_{ji} = 1$, we can infer an upper bound:

$$K_{ij}^{-} \leq K_{ij} \leq 1 - K_{ji}^{-}. \tag{10}$$

The objective of the problem is to find an assignment of all the occurrence shifts: in other words, determining a sequence of the operations mapped to the same machine such that the cycle time is minimum. Note that, once the occurrence shifts are determined, the minimum cycle time can be obtained

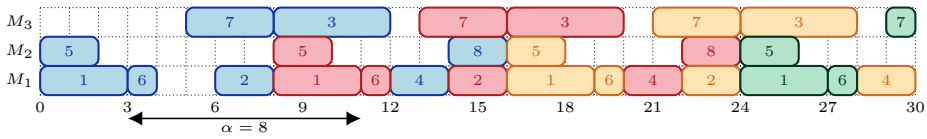


Figure 3: An optimal periodic schedule associated with the CJSP instance with $WIP = 2$.

by computing the critical circuit of the associated graph G since the resulting problem is nothing less than a BCSP.

2.2.1. Robust version

Unlike the BCSP, a robust version of the Cyclic Job Shop Problem has not been considered in the literature. In the following, we present a robust version of the CJSP where the processing times of tasks belongs to the uncertainty set $\mathcal{U}^{\Gamma}$ presented in the Section 2.

Example 2. *Figure 4 illustrates the disjunctive graph G associated to the robust version of the cyclic job shop problem described in Example 2. For the sake of clarity, we don't represent the disjunctive edges in this figure. Unlike the deterministic CJSP, the length of an arc (i, j) belong to an interval $[\bar{p}_i, \bar{p}_i + \hat{p}_i]$.*

Let us consider three different schedules s_1, s_2 and s_3 described by Table 2. All three schedules lead to a cycle time of 8 in nominal conditions (no deviation of the processing times) and are depicted in Fig. 5. However, these schedules don't behave the same way when uncertainties occur. More precisely, considering $\Gamma = 1$, schedule s_1 leads to a cycle time of 13 when $\xi_8 = 1$. In the same conditions, schedule s_3 leads to a cycle time of 12 while schedule s_2 has a cycle time of 11 for his one deviation worst-case ($\xi_3 = 1$). Furthermore, increasing the uncertainty budget to $\Gamma = 2$ lead to a cycle time of 17 (resp. 14 and 14) for schedule s_1 (resp. s_2 and s_3).

Among these 3 schedules, we can conclude that s_2 is the most robust one when considering $\Gamma = 1$ and both s_2 and s_3 are the most robust when $\Gamma = 2$ (but with different worst-case scenario). All these results are resumed in the last columns of Table 2 and in the Figures 6 and 7.

The problem we adress in this study can be casted as follows:

	K_{12}	K_{14}	K_{16}	K_{24}	K_{26}	K_{46}	K_{58}	K_{37}	$\alpha_{\Gamma=0}$	$\alpha_{\Gamma=1}$	$\alpha_{\Gamma=2}$
s_1	0	-1	0	0	1	2	0	1	8	13	17
s_2	0	-1	0	-1	1	2	-1	1	8	11	14
s_3	0	-1	0	0	0	1	-1	0	8	12	14

Table 2: Three schedules of Example 1 and their worst-case cycle time for different value of Γ.

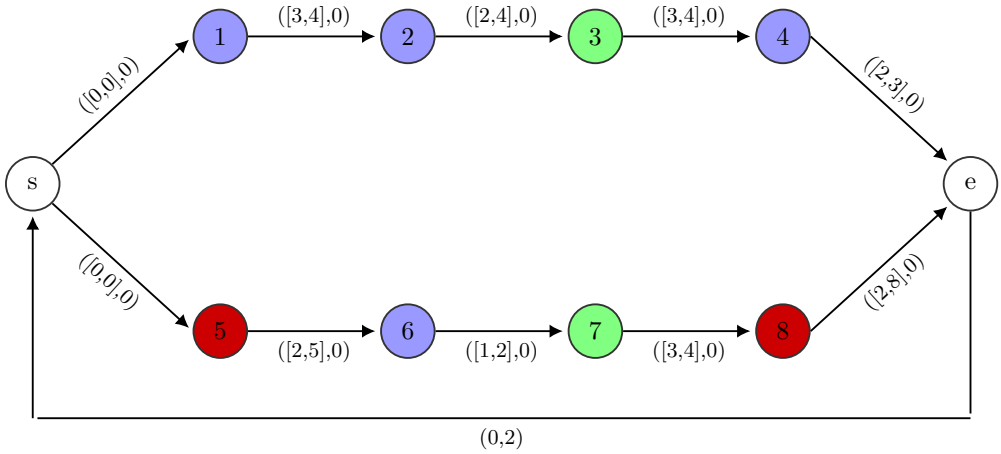


Figure 4: Disjunctive graph associated to instance of Example 2.

$$\min \quad \alpha \tag{11}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq \bar{p}_i + \hat{p}_i \xi_i \quad \forall (i,j) \in \mathcal{P}, \forall \xi \in \mathcal{U}^\Gamma \tag{12}$$

$$t_j(\xi) - t_i(\xi) + \alpha K_{ij} \geq \bar{p}_i + \hat{p}_i \xi_i \quad \forall (i,j) \in \mathcal{D}, \forall \xi \in \mathcal{U}^\Gamma \tag{13}$$

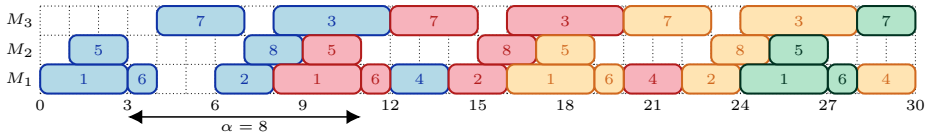
$$K_{ij} + K_{ji} = 1 \quad \forall (i,j) \in \mathcal{D} \tag{14}$$

$$K_{ij}^- \leq K_{ij} \leq 1 - K_{ji}^- \quad \forall (i,j) \in \mathcal{D} \tag{15}$$

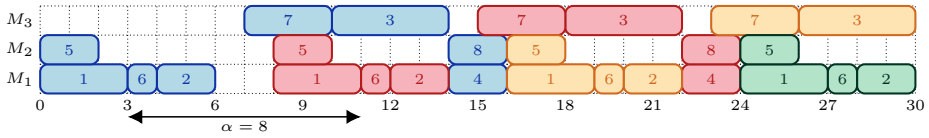
$$K_{ij} \in \mathbb{Z} \quad \forall (i,j) \in \mathcal{D} \tag{16}$$

$$\alpha \geq 0 \tag{17}$$

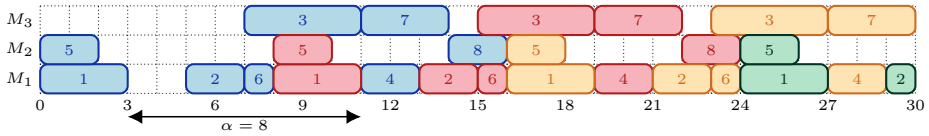
In other words, we aim to find the minimal cycle α and occurrence shifts $(K_{ij})_{(i,j)\in\mathcal{D}}$ such that, for each possible scenario ξ, there always exists a feasible vector of starting time $(t_i(\xi))_{i\in\mathcal{T}}$.



(a) Schedule s_1.

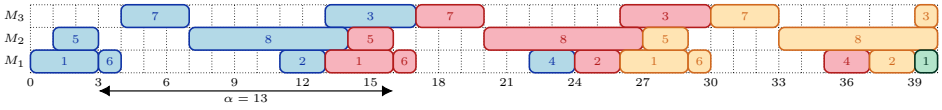


(b) Schedule s_2.

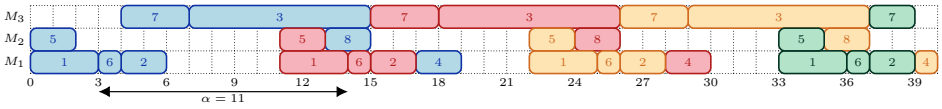


(c) Schedule s_3.

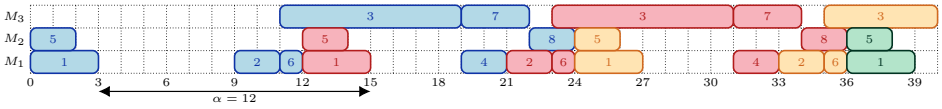
Figure 5: Schedules s_1, s_2 and s_3 in nominal conditions.



(a) Schedule s_1 when $\xi_8 = 1$.

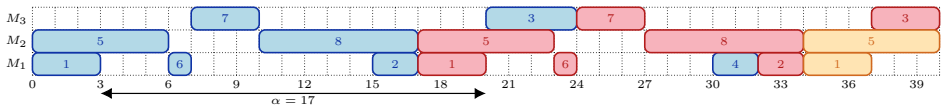


(b) Schedule s_2 when $\xi_3 = 1$.

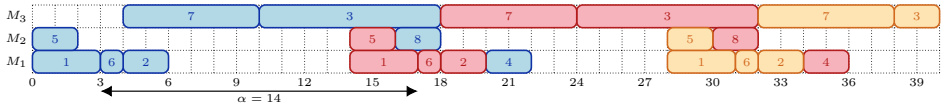


(c) Schedule s_3 when $\xi_3 = 1$.

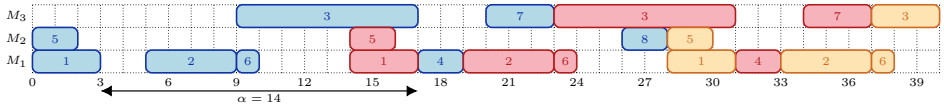
Figure 6: Schedules s_1, s_2 and s_3 in their worst-case scenario with $\Gamma = 1$.



(a) Schedule s_1 when $\xi_5 = 1$ and $\xi_8 = 1$.



(b) Schedule s_2 when $\xi_3 = 1$ and $\xi_7 = 1$.



(c) Schedule s_3 when $\xi_2 = 1$ and $\xi_3 = 1$.

Figure 7: Schedules s_1, s_2 and s_3 in their worst-case scenario with $\Gamma = 2$.

This problem is nonlinear because of the product between the variables $(K_{ij})_{(i,j)\in\mathcal{D}}$ and α but it can be linearized by a variable change first introduced by Hanen [10], by defining $\tau = 1/\alpha$ and $u_i = \tau \times t_i$ for each task i in the set $\mathcal{T}$. The remaining problem is a mixed integer linear program that will be referred as $\mathcal{U}^\Gamma$-CJSP and can be written as follows:

$$\max \quad \tau \tag{18}$$

$$s.t. \quad u_j(\xi) - u_i(\xi) \geq \tau(\bar{p}_i + \hat{p}_i\xi_i) - H_{ij} \qquad \forall\,(i,j) \in \mathcal{P}, \forall \xi \in \mathcal{U}^\Gamma \tag{19}$$

$$u_j(\xi) - u_i(\xi) \geq \tau(\bar{p}_i + \hat{p}_i\xi_i) - K_{ij} \qquad \forall\,(i,j) \in \mathcal{D}, \forall \xi \in \mathcal{U}^\Gamma \tag{20}$$

$$K_{ij} + K_{ji} = 1 \qquad \forall\,(i,j) \in \mathcal{D} \tag{21}$$

$$K_{ij}^- \leq K_{ij} \leq 1 - K_{ji}^- \qquad \forall\,(i,j) \in \mathcal{D} \tag{22}$$

$$K_{ij} \in \mathbb{Z} \qquad \forall\,(i,j) \in \mathcal{D} \tag{23}$$

$$\tau \geq 0 \tag{24}$$

Note that, once the occurrence shifts are fixed, the problem can be solved as a robust BCSP by using the algorithms described in [9].

3. Solution methods for $\mathcal{U}^{\Gamma}$-CJSP

In this section, we present three new methods to solve the $\mathcal{U}^{\Gamma}$-CJSP. The first one is a tailored Branch-and-Bound based on the robust version of the BCSP (see §2.1.1). The two other approaches are based on decomposition methods and more classical in the area of robust optimization.

3.1. Branch-and-Bound method

In the Branch-and-Bound algorithm that we propose to solve, each node of the Branch-and-Bound corresponds to a subproblem defined by the subgraph $G_s = (\mathcal{T}, \mathcal{P} \cup \mathcal{D}_s)$, where $\mathcal{D}_s \subseteq \mathcal{D}$ is a subset of occurrence shifts already fixed. The algorithm starts with a root node G_{root} where $\mathcal{D}_{root} = \emptyset$, in other words, no occurrence shifts are yet decided. The branching is performed by fixing an undetermined occurrence shift K_{ij} and creates a child node for each possible value of K_{ij} in $[K_{ij}^-, 1 - K_{ji}^-]$. Each of these nodes is evaluated by computing the associated cycle time, such that a schedule exists for each scenario ξ. This evaluation is made by means of the robust version of the Howard's algorithm. Our method explores the search tree in best-first search (BeFS) manner, and, in order to branch, it chooses the node having the smallest lower bound. This search strategy can lead to a good feasible solution. A feasible solution is reached when all occurrence shifts are determined. Note that the nominal starting times (i.e. the starting times when no deviation occurs) can be determined by computing the longest path in the graph G where each arc (i, j) is valued by $p_i - \alpha H_{ij}$, and the adjustment is accomplished by shifting the starting time of the following tasks by the value of the deviation. More details are provided in the next subsections.

3.1.1. Computation of an initial upper bound of the cycle time

In order to compute an initial upper bound, we design a heuristic that combines a greedy algorithm with a local search. The greedy algorithm assigns randomly a value to a given occurrence shift K_{ij} in the interval $[K_{ij}^-, 1 - K_{ji}^-]$, and updates the bounds on the rest of the occurrences shifts such that the graph remains consistent. These two operations are repeated

until all occurrence shifts are determined. Once all occurrence shifts are decided, a feasible schedule is obtained, consequently the associated optimal cycle time represents an upper bound of the global optimal cycle time. The local search algorithm consists in improving the (greedy-computed) cycle time by adjusting the values of the occurrence shifts that belong to the critical circuit. The idea behind these improvements is justified by the following proposition:

Proposition 1. *Let $(K_{ij})_{(i,j)\in\mathcal{D}}$ be a vector of feasible occurrence shifts and $\bar{\alpha}$ the associated cycle time given by the critical circuit c. Let $(u,v) \in \mathcal{D}$ be a disjunctive arc such that $(u,v) \in c$. If the following relation holds:*

$$\max_{l\in P^{uv}} \max_{\xi} \sum_{(i,j)\in l} (\bar{p}_i + \hat{p}_i\xi_i) - \bar{\alpha}H_{ij} + (\bar{p}_v + \hat{p}_v\xi_v) - \bar{\alpha}(K_{vu} - 1) \leq 0, \quad (25)$$

where P^{uv} is the set of paths from u to v, then the solution $(K^{'}_{ij})_{(i,j)\in\mathcal{D}}$ where $K'_{uv} = K_{uv} + 1$ and $K'_{vu} = K_{vu} - 1$ has a cycle time less than or equal to $\bar{\alpha}$.

Proof. Let $(K_{ij})_{(i,j)\in\mathcal{D}}$ be a vector of feasible occurrence shifts, $\bar{\alpha}$ the associated cycle time given by the critical circuit c and $(u,v) \in \mathcal{D}$ a disjunctive arc that belongs to c. Let us assume that relation (25) is verified. It is easily seen that putting $K'_{uv} = K_{uv} + 1$ makes the height of the circuit c increase by one and consequently makes the value of its circuit ratio decrease. In order to maintain the condition $K'_{uv} + K'_{vu} = 1$ verified, increasing the value of K_{uv} by 1 involve decreasing the value of K_{vu} by 1. Now, it follows that decreasing the value of K_{vu} by 1 must ensure that the values of the circuits passing through the disjunctive arc (u,v) do not exceed $\bar{\alpha}$. This condition is verified, because by (25) we have:

$$\max_{l\in P^{uv}} \frac{\max\limits_{xi} \sum_{(i,j)\in l} (\bar{p}_i + \hat{p}_i\xi_i) + (\bar{p}_v + \hat{p}_v\xi_v)}{\sum_{(i,j)\in l} H_{ij} + (K_{vu} - 1)} \leq \bar{\alpha}$$

In other words, the maximum circuit ratio passing by the disjunctive arc (j,i) has a value less than or equal to $\bar{\alpha}$. Moreover, since the value of $\bar{\alpha}$ and the values of the processing times are positive, then $\sum_{(i,j)\in l} H_{ij} + (K_{vu} - 1) > 1$. This ensure that the associated graph to the robust CJSP is still consistent and the solution $(K^{'}_{ij})_{(i,j)\in\mathcal{D}}$ is feasible. □

The pseudo-code of the proposed heuristic is given in Algorithm 1.

Algorithm 1 Initial upper bound computation

Compute a lower bounds on the occurrences shifts K_{ij};
for all $(i,j) \in \mathcal{D}$ **do**
    Update bounds on the occurrence shifts;
    Affect randomly value to K_{ij} on the interval $[K_{ij}^-, 1 - K_{ji}^-]$;
end for
Compute the associated cycle time $\bar{\alpha}$ and the critical circuit c.
while $it < it_{max}$ **do**
    Let $(u,v) \in \{(u,v) \in \mathcal{D}$ such that $(u,v) \in c\}$;
    $l_{uv} \leftarrow \max_{l \in P^{uv}} \max_{xi} \sum_{(i,j) \in l} \bar{p}_i + \hat{p}_i \xi_i - \bar{\alpha} H_{ij}$;
    if $l_{uv} + p_v - \bar{\alpha}(K_{vu} - 1) \leq 0$ **then**
        $K_{uv} \leftarrow K_{uv} + 1$;
        $K_{vu} \leftarrow K_{vu} - 1$;
    end if
    Compute the associated cycle time $\bar{\alpha}$ and the critical circuit c;
    it$\leftarrow$ it+1;
end while

3.1.2. Lower bound

In the Branch-and-Bound algorithm, an initial lower bound is derived and compared to the incumbent. If the value of the initial lower bound and the value of the incumbent are equal, then an optimal solution is obtained and the Branch-and-Bound is stopped. It is easily seen that the problem where the disjunctive arcs are ignored is a relaxation of the initial problem. Consequently, the associated cycle time, α_{basic}, is a lower bound on the optimal cycle time. Furthermore, another lower bound can be computed by reasoning on the machine charges. Let $M_{(i)} \in \mathcal{M}$ be a given machine and $S_{(i)} \subseteq \mathcal{T}$ the set of operations mapped on the machine $M_{(i)}$, then the optimal cycle time is such that $\alpha_{opt} \geq \sum_{i \in S_{(i)}} \bar{p}_i + \hat{p}_i \xi_i$, for each ξ. Since this relation is verified for each machine, one can deduce the following lower bound:

$$\alpha_{machine} = \max_{m \in \mathcal{M}, \xi \in \mathcal{U}^\Gamma} \left\{ \sum_{i \in \mathcal{T} : M_{(i)} = m} \bar{p}_i + \hat{p}_i \xi_i \right\}.$$

In the Branch-and-Bound procedure, we set the initial lower bound LB to the maximum value between $\alpha_{machine}$ and α_{basic}.

3.1.3. Node evaluation

In the Branch-and-Bound algorithm, we aim to find a feasible vector $(K_{ij})_{(i,j)\in\mathcal{D}}$ of occurrence shifts such that the value of the associated cycle time is minimal and ensures the existence of a schedule for each $\xi \in \mathcal{U}^{\Gamma}$. In order to fathom a node with a partial solution in the search tree, the node has to be evaluated by computing an associated lower bound. Let us consider a given node of the search tree defined by the subgraph $G_s = (\mathcal{T}, \mathcal{P} \cup \mathcal{D}_s)$, where $\mathcal{D}_s \subseteq \mathcal{D}$ is the set of fixed occurrence shifts. This subgraph represents a relaxation of the initial problem since only a subset of disjunctive arcs is considered. Consequently, the associated cycle time is a lower bound on the optimal cycle time.

3.1.4. Branching scheme and branching rule

To our knowledge, two branching schemes have been proposed in the literature for the cyclic job shop problem. In both branching schemes, the branching is performed on the unfixed occurrence shifts. The first one is introduced in [10]. Based on the interval of possibles values $[K_{ij}^{-}, 1 - K_{ji}^{-}]$ for the occurrence shift K_{ij} such that $(i, j) \in \mathcal{D}$, the author uses a dichotomic branching. In the first generated node, the interval of possible values of the occurrence shifts K_{ij} is restricted to $[K_{ij}^{-}, c_{ij}]$ and in the second one it is restricted to $[c_{ij} + 1, 1 - K_{ji}^{-}]$, where c_{ij} is the middle of the initial interval. The second branching scheme is introduced in [7]. The branching consists in selecting an unfixed disjunction and generates a child node for each possible value of the occurrence shift K_{ij} in the interval $[K_{ij}^{-}, 1 - K_{ji}^{-}]$. In each node, the algorithm assigns the corresponding possible value to the occurrence shift K_{ij}. In this paper, we follow the same branching scheme introduced in [7]. This branching scheme allows us to have, at each node, a subproblem which corresponds to a robust BCSP. Consequently, we can use the existing robust version of the Howard's algorithm to find the cycle time ensuring, for each scenario ξ, the existence of a schedule. Different branching rules

have been tested and numerical tests show that branching on occurrence shifts K_{ij} where $K_{ij}^- + K_{ji}^-$ is maximum yields best running times. This performance can be explained by the fact that this branching rule generates a small number of child nodes, which limits the size of the search tree.

3.2. Decomposition methods for the $\mathcal{U}^\Gamma$-CJSP

This subsection is dedicated to the adaptation of classical decomposition methods for two-stage robust optimisation to the $\mathcal{U}^\Gamma$-CJSP case. More precisely, we adapt row generation and row-and-column generation algorithms described in [1].

Obtaining a solution for the $\mathcal{U}^\Gamma$-CJSP is equivalent to solving the probably large-scale MIP (18)-(24), enumerating all variables and constraints for each scenario. This approach is often unrealistic. That is where decomposition methods make sense and often perform well.

The following proposition characterises the adversarial separation problem of a given assignment for the occurrence shifts.

Proposition 2. *Let $\alpha^* \in \mathbb{R}+$ and $(K_{ij}^*)_{(i,j)\in\mathcal{D}} \in \mathbb{Z}^{|\mathcal{D}|}$ respectively a fixed cycle time and a vector of fixed occurrence shifts. Then, the cycle time α^* is feasible for each scenario $\xi \in \mathcal{U}^\Gamma$ if and only if the value of the optimal solution of the following mixed integer linear program :*

$$\max \quad \sum_{e\in\mathcal{P}} (\bar{p}_e - H_e\alpha^*)u_e + \sum_{e\in\mathcal{D}} (\bar{p}_e - K_e^*\alpha^*)u_e + \sum_{e\in\mathcal{P}\cup\mathcal{D}} \hat{p}_e s_e \tag{26}$$

$$s.t. \quad \sum_{i\in\mathcal{T}} \xi_i \leq \Gamma \tag{27}$$

$$\sum_{e\in\sigma^-(i)} u_e - \sum_{e\in\sigma^+(i)} u_e = 0 \qquad \forall\, i \in \mathcal{T} \tag{28}$$

$$s_e \leq u_e \qquad \forall\, e \in \mathcal{P} \tag{29}$$

$$s_e \leq \xi_e \qquad \forall\, e \in \mathcal{P} \tag{30}$$

$$1 \geq u_e \geq 0 \qquad \forall\, e \in \mathcal{P} \tag{31}$$

$$1 \geq s_e \geq 0 \qquad \forall\, e \in \mathcal{P} \tag{32}$$

$$\xi_i \in \{0, 1\} \qquad \forall\, i \in \mathcal{T} \tag{33}$$

is non-positive.

PROOF. Once the occurrence shift variables is determined, the problem corresponds to the $\mathcal{U}^{\Gamma}$-BCSP that is solved in [9].

Both decomposition algorithms are based on the separation problem (26)-(33). The idea behind these algorithms is to solve a so-called *master problem* which corresponds to the extended formulation of $\mathcal{U}^{\Gamma}$-CJSP (see equations (11)-(17)) by considering only a a subset of scenarios $\mathcal{S} \subset \mathcal{U}^{\Gamma}$. The separation problem give us one scenario $\xi \in \mathcal{U}^{\Gamma}$ for which the solution $(\alpha^*, (K^*_{ij})_{(i,j)\in\mathcal{D}})$ is infeasible, and a dual vector $(U_e)_{e\in\mathcal{P}}$. The difference between the two algorithms is in the way the information is incorporated in the master problem in order to consider the violated scenario. The master problem is formulated as follows:

$$\min \quad \alpha \tag{34}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq \bar{p_i} + \hat{p_i}\xi \quad \forall\,(i,j) \in \mathcal{P}, \xi \in \mathcal{S} \tag{35}$$

$$t_j(\xi) - t_i(\xi) + \alpha K_{ij} \geq \bar{p_i} + \hat{p_i}\xi \quad \forall\,(i,j) \in \mathcal{D}, \xi \in \mathcal{S} \tag{36}$$

$$K_{ij} + K_{ji} = 1 \quad \forall\,(i,j) \in \mathcal{D} \tag{37}$$

$$K^-_{ij} \leq K_{ij} \leq 1 - K^-_{ji} \quad \forall\,(i,j) \in \mathcal{D} \tag{38}$$

$$K_{ij} \in \mathbb{Z} \quad \forall\,(i,j) \in \mathcal{D} \tag{39}$$

Furthermore, the MIP (34)-(39) can be linearized in the same way as in §2.2.1.

3.3. Row generation and Row-and-column generation algorithms

In this subsection, we apply two conventional techniques for solving 2-stage RO problem to our scheduling problem:

- Row Generation (also known as Benders decomposition)
- Row and Column Generation

3.3.1. Row generation algorithm (RG)

The row generation algorithm is similar to Benders decomposition. More precisely, the master problem is first solved, then two cases can arise: either

the value of the objective is non-positive, and in this case the current solution $(\alpha^*, (K^*_{ij})_{(i,j)\in\mathcal{D}})$ of the master is optimal, or the solution is not feasible. In the latter case, we can retrieve the scenario $\xi^* \in \mathcal{U}^\Gamma$ for which the current solution is not feasible and the vector of the dual solution $(u^*_e)_{e\in E\cup\mathcal{D}}$ to generate the following Benders cut:

$$\sum_{e\in\mathcal{P}}(\bar{p}_e + \hat{p}_e\xi^*_e - H_e\alpha)u^*_e + \sum_{e\in\mathcal{D}}(\bar{p}_e + \hat{p}_e\xi^*_e - K_e\alpha)u^*_e \leq 0 \tag{40}$$

where ξ^*_e and u^*_e are given by the separation problem (26)-(33). This is equivalent to force the value of the objective of the separation problem to be non-positive. This would reject the current first stage solution. We note that constraint (40) is not linear, due to the multiplication of the occurrence shift variables $(K_{ij})_{(i,j)\in\mathcal{D}}$ by the cycle time variable α. This product can be linearized by introducing the production rate variable $\tau = \frac{1}{\alpha}$, which gives the following constraint:

$$\sum_{e\in\mathcal{P}}\left(\tau\left(\bar{p}_e + \hat{p}_e\xi^*_e\right) - H_e\right)u^*_e + \sum_{e\in\mathcal{D}}\left(\tau\left(\bar{p}_e + \hat{p}_e\xi^*_e\right) - K_e\right)u^*_e \leq 0 \tag{41}$$

Thus, we have two formulations. For the master problem, to be linear, we must use the formulation with the variable τ. Then, we use the formulation with the variable α for the separation problem. Once the separation problem is solved, we get a cut with the variable α. In order to linearize it and include it in the master problem, we reformulate the cut using the variable τ. The scheme of this algorithm is depicted in Fig. 8.

3.3.2. Column-and-row generation algorithm (CRG)

The column-and-row generation algorithm works in a similar way as the Benders decomposition. The difference between this method and the previous one is the way the information is incorporated into the master problem. Indeed, instead of adding the Benders cut to the master problem, this method adds a block of variables and constraints corresponding to the $\xi^* \in \mathcal{U}^\Gamma$ scenario, which can be formulated as follows:

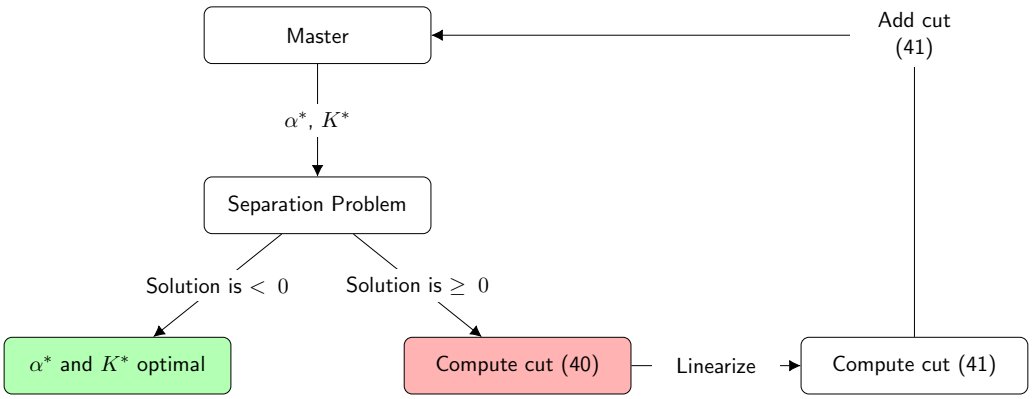


Figure 8: Row generation algorithm scheme

$$t_j(\xi_i^*) - t_i(\xi_i^*) + \alpha H_{ij} \geq \bar{p}_i + \hat{p}_i \xi_i^* \quad \forall\, (i,j) \in \mathcal{P} \tag{42}$$

$$t_j(\xi_i^*) - t_i(\xi_i^*) + \alpha K_{ij} \geq \bar{p}_i + \hat{p}_i \xi_i^* \quad \forall\, s \in \mathcal{M}, \forall\, i,j \in \mathcal{T}_s \tag{43}$$

This block of variables and constraints can be linearized by setting $\tau = \frac{1}{\alpha}$, which gives:

$$u_j(\xi_i^*) - u_i(\xi_i^*) + H_{ij} \geq \tau(\bar{p}_i + \hat{p}_i \xi_i^*) \quad \forall\, (i,j) \in \mathcal{P} \tag{44}$$

$$u_j(\xi_i^*) - u_i(\xi_i^*) + K_{ij} \geq \tau(\bar{p}_i + \hat{p}_i \xi_i^*) \quad \forall\, s \in \mathcal{M}, \forall\, i,j \in \mathcal{T}_s \tag{45}$$

The figure Fig. 9 depicts the scheme of the CRG method.

The global pseudo-code associated to RG and CRG algorithms is given in Algorithm 2.

4. Numerical results

We have implemented the three algorithms in C++ and conducted the numerical experiments on an Intel Xeon E5-2695 processor running at 2.30GHz CPU. The time limit for each instance is set to 900 seconds.

Since there are no existing benchmarks for the CJSP, even in its deter-

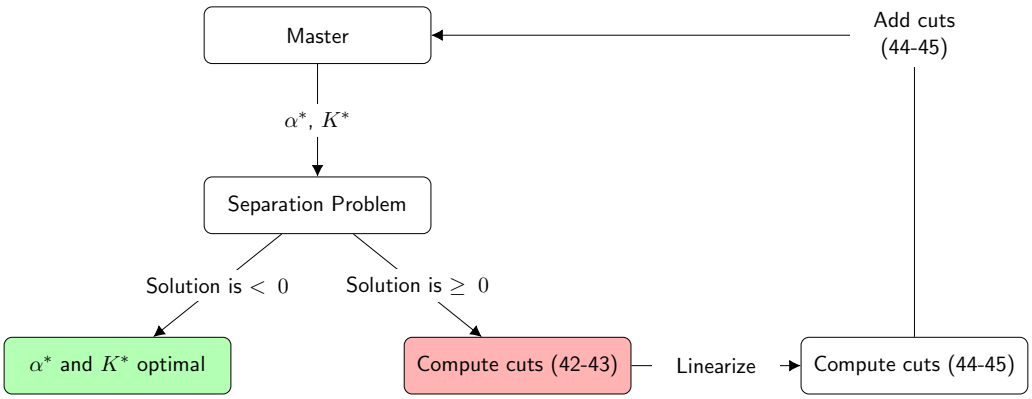


Figure 9: Column-and-row generation algorithm scheme

Algorithm 2 Decomposition algorithms

Input: A disjunctive graph $G = (\mathcal{T}, E)$, uncertainty set $\mathcal{U}^\Gamma$.
Output: A cycle time α such that, for each $\xi \in \mathcal{U}^\Gamma$, there exist a feasible schedule.

repeat
  solve the master problem;
  let $(\alpha^*, (K^*_{ij})_{(i,j)\in\mathcal{D}})$ the optimal solution;
  solve the separation problem, let z^* the value of the objective function;
  if $z^* > 0$ **then**
    add the cut (41) if RG;
    add the cuts (44) and (45) if CRG;
  end if
until $z^* \leq 0$
return α^*

ministic version, we generate new instances. We consider instances in which the number of tasks n belongs to $\{10, 20, 30, 40, 50, 60, 80, 100\}$, the number of jobs j is in $\{2, 3, 4, 5, 6, 10, 16\}$ and the number of machines m is in $\{5, 6, 8, 10\}$. Each nominal duration $\bar{p}_i$ of task i is generated by means of an uniform distribution in $[1, 10]$ and its deviation value $\hat{p}_i$ in $[0, 0.5\bar{p}_i]$. For each configuration, we generate randomly 20 instances.

Tables 3-4 report average solution times for the instances having from 10 to 40 tasks and with a budget of uncertainty varying from 0% to 100%.

All these instances are solved by the Branch-and-Bound algorithm before reaching the time limit. The average running times show that the Branch-and-Bound algorithm is not very sensitive to the variation of the budget of uncertainty. Unlike the B&B, both RG and CRG algorithms are sensitive to the budget of uncertainty. The solving time of RG algorithm raises when the budget of uncertainty increases but we can not draw the same conclusion for CRG algorithm. Indeed, a signficant variation in the computation time is observed for this last algorithm but it is difficult to establish a direct connection of this behavior with the the budget of uncertainty. This can be explained by the number of the nodes explored in the Branch-and-Bound tree which can differ from an instance with a given value of Γ to another one. Tables 3-4 also display the percentage of deviation, for a given budget of uncertainty, of the optimal cycle time α_Γ from the nominal optimal cycle time α_{nom}, where all task durations take their nominal values. This percentage of deviation is computed as $\text{Dev}_\alpha = \frac{\alpha_\Gamma - \alpha_{nom}}{\alpha_{nom}}$. The tables show that the percentage of deviations varies from 25.41% to 56.43%. In other words, these deviations represent the percentage of the nominal cycle time that has to be increased in order to protect a schedule against the uncertainty. We remark that the deviations stabilize when the budget of the uncertainty is greater than 20 or 30 percent. This situation occurs probably when the number of arcs of the circuit having the maximum number of arcs is less than Γ. In this case, increasing Γ does not influence the optimal cycle time. The second situation occurs when heights of other circuits than the actual critical circuit c have greater value than the height of c. In this case, increasing the budget of the uncertainty does not make the value of c lower than the others. Comparing the three algorithms, we can remark that the B&B is the only algorithm able to solve all the instances within the time limit. Both RG and CRG algorithms encounter difficulties to solve the 30 tasks instances and even more significant RG algorithm performances start to collapse with the 40_4_8 instances. Both the computation times and the number of solved instances, indicated in Tables 3-4, also consolidate these insights.

Table 5 shows the number of the instances that are solved before reaching

the time limit. These results concern instances having from 50 to 100 tasks. The Branch-and-Bound is not able to solve all these instances in less then 900 seconds. For example, among instances with 80 tasks, 16 jobs and 5 machines, only three instances have been solved. Nevertheless, the two other algorithms perform worse. Note that CRG is still slightly better than RG.

5. Concluding remarks and perspectives

In this paper, we consider the cyclic job shop problem where the task duration is subject to uncertainty and belong to a polyhedral uncertainty set. We model the problem as two-stage robust optimization problem where the cycle time and the execution order of tasks mapped on the same machine have to be decided before knowing the uncertainty, and the starting times of tasks have to be determined after the uncertainty is revealed. Three algorithms are proposed: a dedicated Branch-and-Bound, a Row Generation method and a Row and Column Generation method. The Branch-and-Bound, based on the Basic Cyclic Scheduling Problem, provides the best results and starts to encounter difficulties from 50 tasks. The two other methods are more classical in the framework of robust optimization but collapse when the number of tasks exceeds 30.

[1] Ayoub, J. and Poss, M. (2016). Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239.

[2] Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

[3] Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35–53.

[4] Brucker, P. and Kampmeyer, T. (2008). A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics*, 156(13):2561–2572.

					B&B		RG		RCG	
#Tasks	#Jobs	#Machines	$\Gamma(\%)$	$\text{Dev}_\alpha(\%)$	T(s)	nb_ins	T(s)	nb_ins	T(s)	nb_ins
10	2	5	0	0	0.012	20	0.015	20	0.017	20
			10	25.41	0.012	20	0.068	20	0.059	20
			20	41.89	0.013	20	0.080	20	0.065	20
			30	48.95	0.015	20	0.066	20	0.062	20
			40	51.67	0.017	20	0.070	20	0.055	20
			50	53.18	0.018	20	0.084	20	0.050	20
			70	53.49	0.021	20	0.081	20	0.040	20
			90	53.49	0.025	20	0.072	20	0.042	20
			100	53.49	0.025	20	0.073	20	0.038	20
20	3	6	0	0	0.298	20	0.139	20	0.149	20
			10	33.61	0.213	20	1.454	20	0.708	20
			20	49.49	0.243	20	1.869	20	0.771	20
			30	55.44	0.568	20	1.918	20	0.970	20
			40	56.43	0.299	20	2.207	20	0.747	20
			50	56.43	0.325	20	2.144	20	0.561	20
			70	56.43	0.378	20	2.085	20	0.500	20
			90	56.43	0.399	20	1.618	20	0.497	20
			100	56.43	0.469	20	1.607	20	0.498	20

Table 3: Average solution time in seconds, number of solved instances and relative deviation of the cycle time from the nominal cycle time.

					B&B		RG		RCG	
#Tasks	#Jobs	#Machines	$\Gamma(\%)$	$\text{Dev}_\alpha(\%)$	T(s)	nb_ins	T(s)	nb_ins	T(s)	nb_ins
30	5	8	0	0	22.643	20	1.992	18	1.828	18
			10	38.25	12.008	20	64.388	17	122.044	17
			20	49.03	15.409	20	90.919	17	117.613	18
			30	50.91	66.247	20	240.929	16	54.921	17
			40	50.91	16.309	20	254.992	11	44.884	17
			50	50.91	17.116	20	183.675	9	38.639	17
			70	50.91	13.833	20	274.775	11	7.745	17
			90	50.91	15.754	20	267.008	12	10.695	17
			100	50.91	15.824	20	266.173	12	20.805	17
40	4	8	0	0	138.917	20	4.878	17	24.833	17
			10	37.92	55.922	20	108.488	13	513.052	10
			20	54.46	92.145	20	339.541	8	368.123	8
			30	54.91	96.758	20	289.097	3	363.923	12
			40	54.91	134.444	20	630.480	16	203.747	13
			50	54.91	155.232	20	-	0	80.396	15
			70	54.91	187.208	20	-	0	41.456	15
			90	54.91	204.481	20	673.780	1	68.928	16
			100	54.91	177.245	20	663.180	1	85.241	16

Table 4: Average solution time in seconds, number of solved instances and relative deviation of the cycle time from the nominal cycle time.

				B&B	RG	RCG
#Tasks	#Jobs	#Machines	Γ(%)	nb_ins	nb_ins	nb_ins
50	5	10	0	11	16	15
			10	10	3	0
			20	11	2	1
			30	14	1	2
			40	13	0	3
			50	13	0	5
			70	12	1	0
			90	12	1	7
			100	12	1	7
60	6	10	0	7	4	3
			10	5	0	1
			20	4	0	1
			30	4	0	2
			40	4	0	2
			50	3	0	2
			70	3	0	3
			90	1	0	3
			100	1	0	3
80	8	16	0	8	0	0
			10	8	0	0
			20	4	0	0
			30	2	0	0
			40	2	0	0
			50	2	0	0
			70	2	0	0
			90	0	0	0
			100	0	0	0
100	10	20	0	0	0	0
			10	0	0	0
			20	3	0	0
			30	2	0	0
			40	1	0	0
			50	0	0	0
			70	0	0	0
			90	0	0	0
			100	0	0	0

Table 5: Number of solved instances in less than 900 seconds among 20 instances.

[5] Che, A., Feng, J., Chen, H., and Chu, C. (2015). Robust optimization for the cyclic hoist scheduling problem. *European Journal of Operational Research*, 240(3):627–636.

[6] Dasdan, A. (2004). Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 9(4):385–418.

[7] Fink, M., Rahhou, T. B., and Houssin, L. (2012). A new procedure for the cyclic job shop problem. *IFAC Proceedings Volumes*, 45(6):69–74.

[8] Gondran, M., Minoux, M., and Vajda, S. (1984). *Graphs and Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

[9] Hamaz, I., Houssin, L., and Cafieri, S. (2018). Robust Basic Cyclic Scheduling Problem. *EURO Journal on Computational Optimization*, 6(3):291–313.

[10] Hanen, C. (1994). Study of a np-hard cyclic scheduling problem: The recurrent job-shop. *European journal of operational research*, 72(1):82–101.

[11] Quinton, F., Hamaz, I., and Houssin, L. (2020). A mixed integer linear programming modelling for the flexible cyclic jobshop problem. *Annals of Operations Research*, 285(1):335–352.

[12] Roundy, R. (1992). Cyclic schedules for job shops with identical jobs. *Mathematics of operations research*, 17(4):842–865.

[13] Thiele, A., Terry, T., and Epelman, M. (2009). Robust linear optimization with recourse. *Rapport technique*, pages 4–37.

[14] Zeng, B. and Zhao, L. (2013). Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461.