A mixed integer linear programming modelling for the flexible cyclic jobshop problem

Félix Quinton, Idir Hamaz, Laurent Houssin

Noname manuscript No.
(will be inserted by the editor)

A mixed integer linear programming modelling for the flexible cyclic jobshop problem

Félix Quinton · Idir Hamaz · Laurent Houssin

Received: date / Accepted: date

Abstract This paper addresses the Cyclic Jobshop Problem in a flexible context. The flexibility feature means that machines are able to perform several kinds of tasks. Hence, a solution of the scheduling problem does not only concern the starting times of the elementary tasks, but also the assignment of these tasks to a unique machine. The objective considered in this paper is the minimisation of the cycle time of a periodic schedule. We formulate the problem as a Mixed Integer Linear Problem and propose a Benders decomposition method along with a heuristic procedure to speed up the solving of large instances. It consists in reducing the number of machines available for each task. Results of numerical experiments on randomly generated instances show that the MILP modelling has trouble solving difficult instances, while our decomposition method is more efficient for solving such instances. Our heuristic procedure provides good estimates for difficult instances.

1 Introduction

This paper tackles the Flexible Cyclic Jobshop Scheduling Problem (FCJSP). We propose a Mixed Integer Linear Problem (MILP) formulation for the FCJSP along with a Benders decomposition algorithm adapted for the FCJSP. We also introduce a heuristic procedure to ease the solving of large instances.

In scheduling theory, the Jobshop Scheduling Problem (JSP) refers to a scheduling problem where tasks are grouped in jobs and executed on specific machines. The JSP is standard problem that have been extensively studied (Błażewicz et al. (1996)), and proven to be NP-hard (Garey and Johnson (1979)). Numerous metaheuristics have been proposed to ease the solving of the JSP (Dell'Amico and Trubian (1993), Schutten (1998)). However, it is inefficient when scheduling infinitely many repetitions of the

Félix Quinton, Laurent Houssin
LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France
E-mail: houssin@laas.fr

Idir Hamaz
LIRMM UMR 5506
Université de Montpellier, 34932 Montpelier Cedex 5, France

same set of jobs, as it would require to schedule each job individually, leading to an immense number of variables in the modelling, which would consequently be very hard to solve.

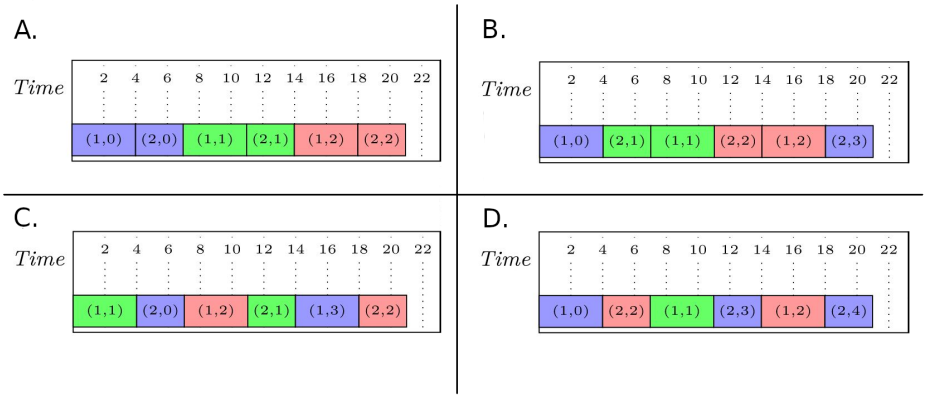
This issue can be solved by modelling the problem as a Cyclic Jobshop Scheduling Problem (CJSP), which consist in indefinitely repeating a given set of activities (Draper et al. (1999), Kim and Lee (2018)). The CJSP has been proven to be NP-complete (Carlier and Chrétienne (1988)), and can be expressed as a MILP (Brucker and Kampmeyer (2008b), Hanen (1994), Roundy (1992), and Serafini and Ukovich (1989)). Two objectives are commonly considered for the CJSP : the minimisation of the cycle time (Hanen (1994)), noted α, which is defined as the time between two iterations of the same task, and the minimisation of the Work In Process (WIP) (Korbaa et al. (2002)), which is the number of occurrences of a job processed simultaneously. In this paper, we will consider the minimisation of the cycle time.

The Flexible Jobshop Scheduling Problem (FJSP) is a generalisation of the JSP, where each task can be executed on several machines with various execution times. This problem can be solved using a MILP (Gomes et al. (2005), Thomalla (2001)) but, due to the complexity of the problem (Xia and Wu (2005)), heuristics approaches such as tabu search (Saidi-Mehrabad and Fattahi (2007)), ant colony optimisation (Rossi and Dini (2007)), or genetic algorithms (Gao et al. (2008)) are very popular.

Despite the richness of the literature about CJSP and FJSP, few papers tackle the FCJSP. Jalilvand-Nejad and Fattahi (2015) proposed a MILP for the FCJSP where the objective is to minimise the total cost of production in an assembly line environment. The authors also provide a simulated annealing algorithm and a genetic algorithm to solve large-scales problems. In Elmi and Topaloglu (2017), the authors proposed an ant colony algorithm to solve a cyclic jobshop robotic cell scheduling problem (CJRSP). In this problem, the realisation of a job requires the objects to be moved from one machine to another by a robot in order to perform the tasks involved in this job. Bożejko et al. (2016) also provide a solution for the CJRSP with two machines and one robot. In Bożejko et al. (2017), a parallel algorithm is used to determine the cycle time of a FJSCP in a graph modelling and the uathors proved several properties on the relationship between the optimal cycle time and the characteristics of the paths in the graph. Zhang et al. (2015) proposed a Mixed Integer Problem (MIP) formulation for the FCJSP with time window constraints and resource capacity constraints.

This paper is organized as follows. In section 2, we introduce the Basic Cyclic Scheduling Problem (BCSP). The basic notations of cyclic scheduling and a linear model for the BCSP are given, and an example is introduced. Section 3 presents the Cyclic Jobshop Scheduling Problem (CJSP). We introduce the notion of machines and resource constraints, and present a linear modelization for the CJSP proposed by Hanen (1994). In Section 4 we introduce flexible machines i.e. machines able to perform several different tasks and define the Flexible Cyclic Jobshop Scheduling Problem (FCJSP). We propose a MILP for this problem and discuss the addition of several set of constraints to speed up the solving. We also propose a Benders decomposition algorithm adapted to the FCJSP and a heuristic procedure to tackle long solving time for large instances. Numerical instances are presented in Section 5.

Fig. 1 Gantt diagrams displaying feasible schedules for tasks 1 and 2 depending on the height of precedence constraint $C = (1, 2, 4, H_{1,2})$: A. $H_{1,2} = 0$, B. $H_{1,2} = 1$, C. $H_{1,2} = -1$, D. $H_{1,2} = 2$



2 Basic cyclic scheduling problem

The Basic Cyclic Scheduling Problem (BCSP) is a generalisation of the basic scheduling problem where a schedule of elementary tasks $i \in \mathcal{T} = \{1, ..., n\}$ is infinitely repeated. This problem has been studied a lot since it has applications in many fields such as parallel processing (Hanen and Munier (1995)), staff scheduling (Karp and Orlin (1981)), or robotic scheduling (Ioachim and Soumis (1995),Kats and Levner (1996)). Hamaz et al. (2018a) studied the a BCSP where the processing times are affected by an uncertainty effect.

To specify a BCSP, we are given a set of elementary tasks $i \in \mathcal{T}$ and their respective execution time p_i. Also, we define $t_i(k) \geq 0$ the starting time of occurrence k of task $i \in \mathcal{T}$. The order in which the tasks are to be executed is fixed by precedence constraints. A precedence constraint indicating that task i must be completed before starting the execution fo task j can be expressed as a quadruplet $(i, j, p_i, H_{i,j})$, where $H_{i,j}$ is called the height of the precedence constraint and represents the occurrence shift between tasks i and j. In simple terms, it states that the $k + H_{i,j}$-th occurrence of task j cannot start before the end of the k-th occurrence of task i. Mathematically, it states that :

$$t_j(k + H_{i,j}) \geq p_i + t_i(k).$$

Example 1 : Let us consider the precedence constraint $C = (1, 2, 4, H_{1,2})$. Also, we fix $p_2 = 3$. If $H_{1,2} = 0$, it follows from C that $t_2(k) \geq 4 + t_1(k)$. Then a feasible schedule for task 1 and 2 is displayed in Figure 1 A. Note that in Figure 1, the label of each task shows the number of the task and the occurrence it belongs to. If $H_{1,2} = 1$, it follows from C that $t_2(k+1) \geq 4+t_1(k)$. A feasible schedule for task 1 and 2 is showed in Figure 1 B. Now let us consider the case $H_{1,2} = -1$. From C we can derive that $t_2(k-1) \geq 4 + t_1(k)$. A feasible schedule for task 1 and 2 is displayed in Figure 1 C. Finally, Figure 1 D. displays a feasible schedule if $H_{1,2} = 2$.

The BCSP can be represented as an oriented weighed graph where each vertex represent an elementary task $i \in \mathcal{T}$ and the weight of the edges are the couple (p_i, H_{ij})

such as the set of edges $\mathcal{E}$ reproduce the set of precedence constraints. We add fictive tasks s end e to represent the start and the end of an occurrence.

Let us denote $t_i = t_i(0), i \in \mathcal{T}$. In this study, the objective considered is the minimisation of the cycle time α, which is the time between two occurrences of the same task. Consequently, it is defined by the equations :

$$t_i(k) = \alpha k + t_i \quad \forall i \in \mathcal{T}.$$

There are two methods to solve the BCSP. The first is based on graph and consists in finding the maximum mean cycle in the graph i.e the cycle C maximising

$$\alpha(C) = \frac{\sum_{(i,j)\in C} p_i}{\sum_{(i,j)\in C} H_{ij}}.$$

This cycle is called critical path and an application to BSCP has been proposed by Levner and Kats (1998). This method has also been used by Chretienne et al. (1991) to solve a BCSP with deadlines,

The second method consist in using mathematical programming. The BCSP can be expressed as a linear problem as follows (Kampmeyer (2006)).

$$\min \alpha$$

s.t.

$$\alpha \geq p_i, \quad \forall i \in \mathcal{T} \tag{1a}$$
$$t_j + \alpha H_{i,j} \geq t_i + p_i, \quad \forall (i,j) \in \mathcal{E} \tag{1b}$$
$$t_i \geq 0, \quad \forall i \in \mathcal{T}. \tag{1c}$$

In this modelling, the set of constraints (1a) correspond to non-reentrance constraints which ensure that the k-th occurence of any elementary task $i \in \mathcal{T}$ always precedes its $k+1$-th occurence. Constraints (1b) express the precedence constraints.

Example 2 : Table I reports the data of a BCSP instance with 7 elementary tasks. The graph representing this example is presented in Figure 2. After solving this BCSP, we find an optimal cycle time of $\alpha = 9.5$. A Gantt diagram displaying the optimal solution of the BCSP instance is presented in Figure 3.

Table 1 An example for the BCSP

Task	1	2	3	4	5	6	7
Time	5	4	5	5	4	3	5

Fig. 2 Graph representation of the BCSP instance introduced in Example 2.

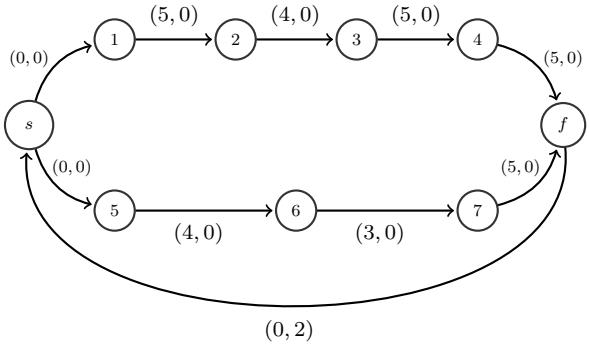
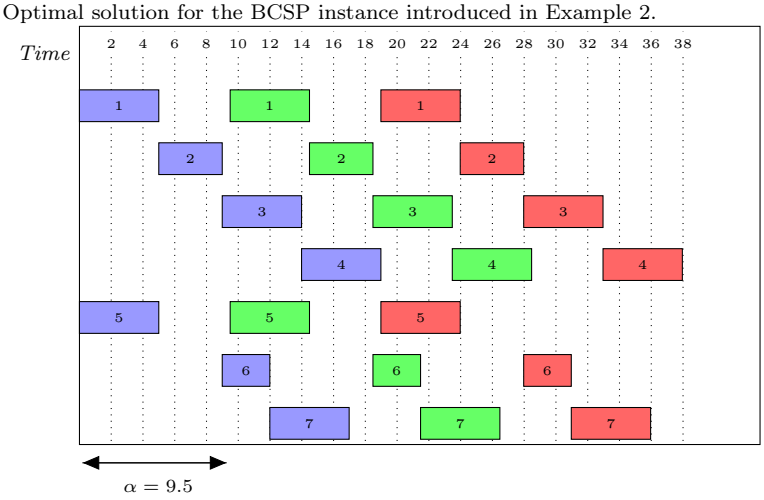


Fig. 3 Optimal solution for the BCSP instance introduced in Example 2.



3 Cyclic jobshop scheduling problem

In the cyclic jobshop scheduling problem (CJSP), each elementary task $i \in \mathcal{T}$ is assigned to a machine $r_i \in \mathcal{R} = \{1, ..., R\}$, with $R < n$. In the CJSP, elementary tasks linked by precedence constraints constitute jobs. For instance, in a manufacturing context, a job might represent the manufacturing process of a product as a whole, while an elementary task represents only one step of the manufacturing process.

Due to its numerous applications in large-scale production scheduling, the CJSP has received at lot of attention from the research community. Hanen (1994) proposed a MILP for this problem and proved some of its properties.Brucker et al. (2012) also proposed a MILP for a CJSP with transportation by a single robotic cell. Draper et al. (1999) proposed an alternative formulation to solve CJSP problems as constraint satisfaction problems. Hamaz et al. (2018b) studied the CJSP with uncertain processing time. Houssin (2011) proposed an exact method to solve a CJSP using (max, plus) algebra. Also, several methods have been proposed to solve a CJSP with blocking,

such as Petri-net modelling (Song and Lee (1998)), tabu search algorithm (Brucker and Kampmeyer (2008a)), and an hybrid algorithm based on particle swarm optimisation and simulated annealing (Jamili et al. (2011)).

The inclusion of machines in the CJSP leads to a lack of resources, since the tasks are competing for the working time of the machines. This lack of resources is represented by disjunction constraints, which state that for a pair of tasks $(i,j) \in \mathcal{T}^2, i \neq j$, that must be executed on the same machine, i.e. $r_i = r_j$, an occurrence of tasks i and an occurrence of j, cannot be executed at the same time. In the following of this paper, we will denote by $\mathcal{D} = \{(i,j)|R(i) \cap R(j) \neq \emptyset\}$ the set of pairs of tasks linked by disjunction constraints.

Mathematically, the disjunction between two tasks $(i,j) \in \mathcal{T}^2, i \neq j$ is modelled with the two following disjunction constraints (2).

$$\begin{cases} t_j(k + K_{i,j}) \geq t_i(k) + p_i \\ t_i(k + K_{j,i}) \geq t_j(k) + p_j. \end{cases} \tag{2}$$

Where K_{ij} (resp. K_{ji}) is the height of the disjunction constraint, i.e. the occurrence shift between task i and j (resp. j and i). It has been proven by Hanen (1994) that a feasible schedule for a CJSP must satisfy :

$$K_{ij} + K_{ji} = 1 \quad \forall (i,j) \in \mathcal{D}. \tag{3}$$

Note that in this problem the variables are the cycle time α, the starting times of each elementary tasks $(t)_{i \in \mathcal{T}}$, and the heights of the disjunctive constraints, $(K)_{(i,j) \in D}$.

A feature of the CJSP is the Work In Process (WIP). It corresponds to the maximum number of occurrences of a job processed simultaneously. Mathematically, the role of the WIP can be modelled as the height of the precedence constraint from fictive task e to fictive task s, and can be explained by the equation :

$$s(k) \geq e(k - WIP).$$

Some papers focused on the minimisation of the WIP (Hillion and Proth (1989), Korbaa et al. (2002)). In these papers, the heights of the precedence constraints $H_{ij}, (i,j) \in \mathcal{E}$ are variables and the objective consist in a linear function of those heights. However in our study, we focus on the minimisation of the cycle time α, so the WIP and the $H_{ij}, (i,j) \in \mathcal{E}$ are given. Modelling of the CJSP for the minimisation of the cycle time α with know heights as been proposed by Hanen (1994) and is used by Brucker et al. (2012) to solve a CJSP with transportation.

A CJSP can be solved by writing the problem as a MILP, which can be solved using mathematical programming, or using a dedicated Branch-and-Bound procedure (Fink et al. (2012), Hanen (1994)).

To introduce the MILP modelling of the CJSP proposed by Hanen (1994), we first introduce a Mixed Integer Non Linear Programming (MINLP) modelling also presented in Hanen (1994):

$$\min \alpha$$

s.t.

$$\alpha \geq p_i, \quad \forall i \in \mathcal{T} \tag{4a}$$
$$t_j + \alpha H_{i,j} \geq t_i + p_i, \quad \forall (i,j) \in \mathcal{E} \tag{4b}$$
$$t_j + \alpha K(i,j) \geq t_i + p_i, \quad \forall (i,j) \in \mathcal{D} \tag{4c}$$
$$K(i,j) + K(j,i) = 1, \quad \forall (i,j) \in \mathcal{D} \tag{4d}$$
$$K(i,j) \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D} \tag{4e}$$
$$t_i \geq 0, \quad \forall i \in \mathcal{T}. \tag{4f}$$

In this model, constraints (4a) are the non-reentrance constraints and constraints (4b) are the precedence constraints. Constraints (4c) express the disjunction constraints as defined in (2), while constraints (4d) express the disjunctive heights property of the CJSP enunciated by Hanen (1994) and introduced in (3). Non linearity appears in constraints (4c) as the product of variables α and $K(i,j), (i,j) \in \mathcal{D}$.

As integer non linear programming is very unpractical, we want to linearize the model. Hanen (1994) proposes to define the variable $\tau = \frac{1}{\alpha}$ and for all $i \in \mathcal{T}$, the variable $u_i = \frac{t_i}{\alpha}$. Then CJSP can then be as a MILP in the following form Hanen (1994):

$$\max \tau$$

s.t.

$$\tau \leq \frac{1}{p_i}, \quad \forall i \in \mathcal{T} \tag{5a}$$
$$u_j + H_{i,j} \geq u_i + \tau p_i, \quad \forall (i,j) \in \mathcal{E} \tag{5b}$$
$$u_j + K(i,j) \geq u_i + \tau p_i, \quad \forall (i,j) \in \mathcal{D} \tag{5c}$$
$$K(i,j) + K(j,i) = 1, \quad \forall (i,j) \in \mathcal{D} \tag{5d}$$
$$K(i,j) \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D} \tag{5e}$$
$$u_i \geq 0, \quad \forall i \in \mathcal{T}. \tag{5f}$$

Example 3 : In Table 2, we introduce an updated version of Example 2 presenting a CJSP with 4 machines. This example is represented as a graph in Figure 4, displaying only the disjunctive constraints for robot M_3 for cleanliness purposes. Note that tasks 1 to 4 form job 1 while tasks 5 to 7 form job 2. Solving this CJSP, we find an optimal cycle time $\alpha = 13$. A Gantt diagramm displaying the optimal solution for this CJSP is presented in Figure 5.

Table 2 An example for the CJSP

Task	1	2	3	4	5	6	7
Time	5	4	5	5	4	3	5
Robot	M_1	M_1	M_3	M_1	M_2	M_4	M_3

Fig. 4 Graph representation of the CJSP instance introduced in Example 3.

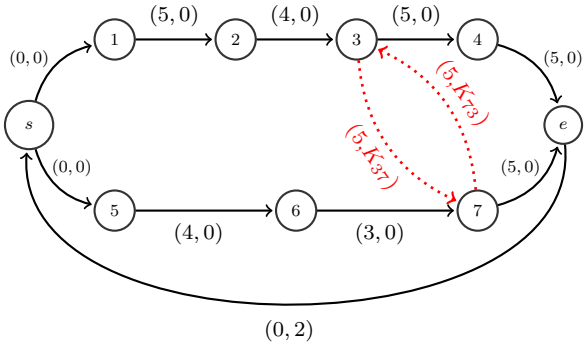
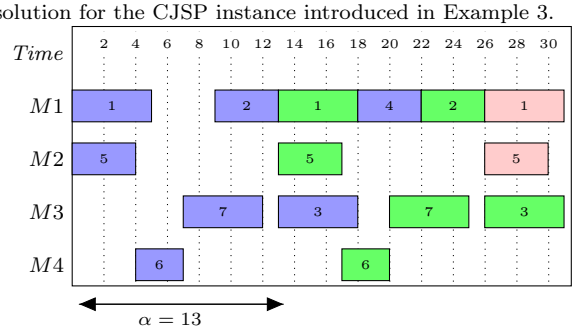


Fig. 5 Optimal solution for the CJSP instance introduced in Example 3.



4 Flexible cyclic jobshop scheduling problem

The flexible cyclic jobshop scheduling problem (FCJSP) is a CJSP where the elementary tasks are flexible. This means that the execution of a task $i \in \mathcal{T}$, is assigned to exactly one machine r in a set of machines that is a subset of the set of machines $\mathcal{R}$ specific to task i. This subset is denoted $R(i) \subset \mathcal{R}$. We model the assignement of a task $i \in \mathcal{T}$ to a machine $r \in R(i)$ as a decision variable $m_{i,r}$ defined as follows :

$$\forall i \in \mathcal{T}, \forall r \in R(i),$$
$$m_{i,r} = \begin{cases} 1 \text{ if task i is assigned to machine r} \\ 0 \text{ otherwise.} \end{cases}$$

Each assignment of a task $i \in \mathcal{T}$ to a machine $r \in R(i)$ is associated with a given execution time denoted p_{ir}. Also, because we do not know *a priori* on which machine each task will be assigned, we do not know the set $(i,j) \in \mathcal{T}^2, i \neq j, R(i,j) \neq \emptyset$ of pairs of tasks which are connected by a disjunctive constraint.

Example 4 : In Table 4, we have updated Example 3 so that it fits the FCJSP. Each task can be assigned to a subset of 3 out of 4 machines. Note that the new possibilities come with a higher execution time.

Table 3 An example for the FCJSP

Task	1			2			3			4		
Robot	M_1	M_2	M_4	M_1	M_2	M_3	M_2	M_3	M_4	M_1	M_2	M_3
Time	5	6	6	4	5	6	7	5	6	4	7	5
Task	5			6			7					
Robot	M_1	M_2	M_3	M_1	M_3	M_4	M_1	M_2	M_3			
Time	4	4	5	3	4	2	5	7	5			

4.1 MIP model

In this section, we propose a MILP modelling for the FCJSP. A first model was proposed in Quinton et al. (2018). Let $R(i,j) = R(i) \cap R(j), \forall i,j \in \mathcal{T}$ the set of machines available both for tasks i and j. Recall that $\mathcal{E}$ denotes the set of precedence constraints and that $\mathcal{D}$ denotes the set of disjunctive constraints.

The MILP for the FCJSP can be formulated as :

$$\max \tau$$

s.t.

$$\tau \leq \sum_{r \in R(i)} \frac{m_{i,r}}{p_{i,r}}, \quad \forall i \in \mathcal{T} \tag{6a}$$

$$u_j + H_{i,j} \geq u_i + \sum_{r \in R(i)} p_{i,r} y_{i,r}, \quad \forall (i,j) \in \mathcal{E} \tag{6b}$$

$$P_1(2 - m_{i,r} - m_{j,r}) + u_j + K_{i,j} \geq u_i + p_{i,r} y_{i,r}, \quad \forall (i,j) \in \mathcal{D}, \forall r \in R(i,j) \tag{6c}$$

$$\sum_{r \in R(i)} y_{i,r} = \tau, \quad \forall i \in \mathcal{T} \tag{6d}$$

$$y_{i,r} \leq m_{i,r}, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{6e}$$

$$y_{i,r} \geq 0, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{6f}$$

$$\sum_{r \in M(i)} m_{i,r} = 1, \quad \forall i \in \mathcal{T} \tag{6g}$$

$$K(i,j) + K(j,i) = 1, \quad \forall (i,j) \in \mathcal{D} \tag{6h}$$

$$\tau \geq 0 \tag{6i}$$

$$u_i \geq 0, \quad \forall i \in \mathcal{T} \tag{6j}$$

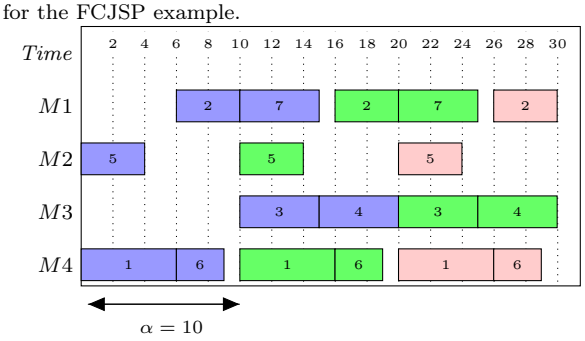
$$K(i,j) \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D} \tag{6k}$$

$$m_{i,r} \in \{0,1\}, \quad \forall i \in \mathcal{T}, \quad \forall r \in R(i). \tag{6l}$$

Constraints (6a) correspond to non-reentrance constraints. They ensure that the k-th occurence of any elementary task $i \in \mathcal{T}$ always precedes its $k+1$-th occurence. As constraints (6g) ensure that exactly one $m_{i,r}$ is equal to 1 for all tasks, these non-reentrance constraints are equivalent to the non-reentrance constraints from the CJSP (5a).

Constraints (6b) set the precedence constraints, i.e. the order in which tasks have to be executed. There is one precedence constraint for each couple of tasks $(i,j) \in \mathcal{E}$. As the assignment of task $i \in \mathcal{T}$ to a machine $r \in R(i)$ set the execution time of

Fig. 6 Solution for the FCJSP example.



i to $p_{i,r}$, we want only the effective execution time to be taken into account in the precedence constraints. To that end, we introduce the $y_{i,r}$ variables, which are defined by constraints (6d) to (6f) such that :

$$\forall i \in \mathcal{T}, \forall r \in R(i),$$
$$y_{i,r} = \begin{cases} \tau \text{ if task i is assigned to machine r} \\ 0 \text{ otherwise.} \end{cases}$$

Indeed, let $i \in \mathcal{T}, r \in R(i)$, if $m_{i,r} = 0$, constraints (6e) and (6f) force $y_{i,r} = 0$. As constraints (6g) ensure that for each task $i \in \mathcal{T}$, there is exactly one $r \in R(i)$ such that $m_{i,r} = 1$, constraints (6d) together with constraints (6e) ensure that the corresponding $y_{i,r}$ has to be equal to τ. Therefore, for any precedence constraint $(i, j) \in \mathcal{E}$, the model only take into account the execution time $p_{i,r}$ if task i is executed on machine $r \in R(i)$.

Constraints (6c) represents the disjunctive constraints, ensuring that two tasks are not executed at the same time period on the same machine. There is one disjunction constraint for each couple of tasks $(i, j) \in \mathcal{D}$ and machine $r \in R(i, j)$, and we want those constraints to be activated only when tasks i and j are assigned to the same machine r as otherwise, the couple of tasks (i, j) does not generate any lack of resources. The P_1 term allows us to deactivate the disjunctive constraints for which tasks i and j are not executed on the same machine $r \in R(i, j)$ i.e. the constraints for which $m_{i,r} + m_{j,r} < 2$. We set :

$$P_1 = \max_{\substack{i \in \mathcal{T} \\ r \in R(i)}} p_{i,r}.$$

Constraints (6g) ensure that every task $i \in \mathcal{T}$ is assigned to exactly one machine $r \in R(i)$. Constraints (6h) state the usual constraints for the CJSP, introduced by Hanen (1994).

*Example 4.1 :*We have solved the Example 4 with the proposed MILP (6). We have obtained a cycle time $\alpha = 10$. Note that the optimal cycle time has decreased even if the new assignment possibilities introduced in Example 4 came with a higher execution time. The solution is displayed as Gantt diagram in Figure 6.

4.2 Benders decomposition

The FCJSP formulated as a MILP (6), can be very hard to solve. Using CPLEX, difficult instances with a large number of tasks or with few robots might exceed any reasonable time limit. To tackle this issue, we propose a Benders decomposition for the FCJSP. In the usual Benders decomposition scheme introduce by Benders (1962), the Master Problem (MP) corresponding to the MILP (6) we proposed for the FCJSP consists in an integer linear problem composed of the constraints from (6) involving only the integer variables, and the optimality cuts generated at each iteration of the Benders algorithm. This master problem can be written as (7).

$$\max z$$

s.t.

$$\sum_{r \in r(i)} m_{i,r} = 1, \quad \forall i \in \mathcal{T} \tag{7a}$$

$$K_{i,j} + K_{j,i} = 1, \quad \forall (i,j) \in \mathcal{D} \tag{7b}$$

$$z + \sum_{\substack{i \in \mathcal{T} \\ r \in R(i)}} m_{i,r} \Big(P_1 \sum_{\substack{(i,j) \in \mathcal{D} \\ r \in R(i,j)}} \left(\pi^d_{i,j,r} + \pi^d_{j,i,r} \right) - \frac{\pi^{nr}_i}{m_{i,r}} - \pi^{Y2}_{i,r} \Big) - \sum_{(i,j) \in \mathcal{D}} K_{i,j} \sum_{r \in R(i,j)} \pi^d_{i,j,r} \leq \sum_{(i,j) \in \mathcal{E}} H_{i,j} \pi^p_{i,j} + 2P_1 \sum_{(i,j) \in \mathcal{D}} \left(\pi^d_{i,j,r} + \pi^d_{j,i,r} \right), \quad \forall \pi \in \mathcal{P} \tag{7c}$$

$$m_{i,r} \in \{0,1\}, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{7d}$$

$$z \in \mathbb{R}, \quad K_{i,j} \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D}. \tag{7e}$$

Where constraints (7c) are the usual Benders optimality cuts. $\mathcal{P}$ is a set of extreme points, each corresponding to an iteration of the algorithm. The generation of new elements for $\mathcal{P}$ is explained by algorithm 4.2. Constraints (7a) and (7b) reproduce constraints (6h) and (6g) from the MILP. However, those constraints are not sufficient to ensure that the solution $\overline{y}$ provided by the MP is feasible for the MILP as $\overline{y}$ could generate negative height cycles (where $\overline{y} = (\overline{m}, \overline{K})$ represents a solution of master problem (7)). In the usual Benders decomposition scheme, this issue is handled with feasibility cuts. However in our case, the Benders algorithm produces a very large number of feasibility cuts which greatly slows its convergence. To avoid this issue we propose cuts (8a) and (8b) to ensure that there exist feasible starting times $\{w_i, i \in \mathcal{T}\}$ fulfilling the disjunction constraints and precedence constraints implied by the integer variables $\{m_{i,r}, \forall i \in \mathcal{T}, \forall r \in R(i)\}$ and $\{K_{i,j}, \forall (i,j) \in \mathcal{D}\}$. Also, we introduce in constraints (8a) and (8b) a lower bound τ_{LB} of the throughput. This lower bound can easily be obtained as it does not need to be qualitative. For instance, one may set $\tau_{LB} = (\sum_{i \in \mathcal{T}} \max_{r \in R(i)} p_{i,r})^{-1}$.

$$P_1(1 - m_{i,r}) + w_j + H_{i,j} \geq w_i + \tau_{LB} p_{i,r}, \quad \forall (i,j) \in \mathcal{E} \tag{8a}$$

$$P_1(2 - m_{i,r} - m_{j,r}) + w_j + K_{i,j} \geq w_i + \tau_{LB} p_{i,r}, \quad \forall (i,j) \in \mathcal{D}, \forall r \in R(i,j) \tag{8b}$$

$$w_i \in \mathbb{R}_+, \quad \forall i \in \mathcal{T}. \tag{8c}$$

The remaining constraints of (6), involving only continuous variables or a combination of continuous and integer variables, compose the primal sub-problem. It can be written as linear problem (9).

$$\max \tau$$

s.t.

$$\tau \leq \sum_{r \in R(i)} \frac{\overline{m}_{i,r}}{p_{i,r}}, \quad \forall i \in \mathcal{T} \tag{9a}$$

$$u_j + H_{ij} \geq u_i + \sum_{r \in R(i)} y_{i,r} p_{i,r}, \quad \forall (i,j) \in \mathcal{E} \tag{9b}$$

$$P_1(2 - \overline{m}_{i,r} - \overline{m}_{j,r}) + u_j + \overline{K}_{i,j} \geq u_i + y_{i,r} p_{i,r}, \quad \forall (i,j) \in \mathcal{D}, \forall r \in R(i,j) \tag{9c}$$

$$\sum_{r \in R(i)} y_{i,r} = \tau, \quad \forall i \in \mathcal{T} \tag{9d}$$

$$y_{i,r} \leq \overline{m}_{i,r}, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{9e}$$

$$\tau > 0, \quad u_i \geq 0, \forall i \in \mathcal{T}, \quad y_{i,r} \geq 0, \forall i \in \mathcal{T}, \forall r \in R(i). \tag{9f}$$

Hence the dual sub-problem is:

$$\min \sum_{i \in \mathcal{T}} \pi_i^{nr} \sum_{r \in R(i)} \frac{\overline{m}_{i,r}}{p_{i,r}} + \sum_{\substack{(i,j) \in \mathcal{D} \\ r \in R(i,j)}} \pi_{i,j,r}^{d} (P_1(2 - \overline{m}_{i,r} - \overline{m}_{j,r}) + \overline{K}_{i,j})$$

$$+ \sum_{(i,j) \in \mathcal{E}} \pi_{i,j}^{p} H_{i,j} + \sum_{\substack{i \in \mathcal{T} \\ r \in R(i)}} \pi_{i,r}^{y2} \overline{m}_{i,r}$$

s.t.

$$\sum_{i \in \mathcal{T}} \pi_i^{nr} - \pi_i^{y1} \geq 1 \tag{10a}$$

$$\sum_{(i,j) \in \mathcal{E}} \pi_{i,j}^{p} - \sum_{(j,i) \in \mathcal{E}} \pi_{j,i}^{p} + \sum_{\substack{(i,j) \in \mathcal{D} \\ r \in R(i,j)}} \pi_{i,j,r}^{d} - \sum_{\substack{(j,i) \in \mathcal{D} \\ r \in R(i,j)}} \pi_{j,i,r}^{d} \geq 0, \quad \forall i \in \mathcal{T} \tag{10b}$$

$$\sum_{(i,j) \in \mathcal{E}} \pi_{i,j}^{p} p_{i,r} + \sum_{(i,j) \in \mathcal{D}} \pi_{i,j,r}^{d} p_{i,r} \geq 0, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{10c}$$

$$\pi_i^{nr} \geq 0, \quad \forall i \in \mathcal{T} \tag{10d}$$

$$\pi_{i,j}^{p} \geq 0, \quad \forall (i,j) \in \mathcal{E} \tag{10e}$$

$$\pi_{i,j,r}^{d} \geq 0, \quad \forall (i,j) \in \mathcal{D}, \forall r \in R(i,j) \tag{10f}$$

$$\pi_i^{y1} \in \mathbb{R}, \quad \forall i \in \mathcal{T} \tag{10g}$$

$$\pi_{i,r}^{y2} \geq 0, \quad \forall i \in \mathcal{T}, \forall r \in R(i). \tag{10h}$$

Where $\pi_i^{nr}, i \in \mathcal{T}$ is the dual variable associated to the non-reentrance constraint for task i, $\pi_{i,j}^{p}, (i,j) \in \mathcal{E}$ is the dual variable associated to the precedence constraint from

task i to task j, $\pi^d_{i,j,r}, (i,j) \in \mathcal{D}, r \in R(i,j)$ is the dual variable associated with the disjunction constraint between tasks i and j for machine r, and π^{y1} (resp. π^{y2}) are the dual variables associated with the set of constraints (9e) (resp. (9f)).

To further enhance our Benders decomposition algorithm, we use the algorithm proposed by Magnanti and Wong (1981) to generate Pareto-optimal cuts. To do so, we need to define the Magnanti-Wong dual sub-problem. Keeping the same notations as for (10), the Magnanti-Wong dual sub-problem can be written as linear problem (11).

In (11), $\overline{y} = (\overline{m}, \overline{K})$ represents a solution of master problem (7), z^* is the value of the objective of the dual sub-problem (10), and $y^0 = (m^0, K^0)$ is a core point. In general, generating a core point is not straightforward. However, Papadakos (2008) showed that given an integer solution $\overline{y}$, $\lambda\overline{y}$ is a core point, for any $\lambda \in]0,1[$. The author also showed that any convex combination of core points is a core point. So, noting y^0_k the core point used at iteration k, and $\overline{y}_k$ the integer solution at iteration k, we propose to use $y^0_k = \frac{1}{2}y^0_{k-1} + \frac{1}{2}\overline{y}_k$ as a core point for stating the Magnanti-Wong dual sub-problem corresponding to the k-th iteration, $k > 0$. For the first iteration, we set $y^0_0 = \frac{1}{2}\overline{y}_0$.

$$\min \sum_{i\in\mathcal{T}} \pi^{nr}_i \sum_{r\in R(i)} \frac{m^0_{i,r}}{p_{i,r}} + \sum_{\substack{(i,j)\in\mathcal{D}\\ r\in R(i,j)}} \pi^d_{i,j,r}(P_1(2 - m^0_{i,r} - m^0_{j,r}) + K^0_{i,j}) + \sum_{(i,j)\in\mathcal{E}} \pi^p_{i,j} H_{i,j} + \sum_{\substack{i\in\mathcal{T}\\ r\in R(i)}} \pi^{y2}_{i,r} m^0_{i,r} \tag{11}$$

s.t.

$$\sum_{i\in\mathcal{T}} \pi^{nr}_i \sum_{r\in R(i)} \frac{\overline{m}_{i,r}}{p_{i,r}} + \sum_{\substack{(i,j)\in\mathcal{D}\\ r\in R(i,j)}} \pi^d_{i,j,r}(P_1(2 - \overline{m}_{i,r} - \overline{m}_{j,r}) + \overline{K}_{i,j}) + \sum_{(i,j)\in\mathcal{E}} \pi^p_{i,j} H_{i,j} + \sum_{\substack{i\in\mathcal{T}\\ r\in R(i)}} \pi^{y2}_{i,r} \overline{m}_{i,r} = z^* \tag{11a}$$

and

$$\text{(10a)-(10h)}.$$

Having defined the Magnanti-Wong dual sub-problem and an algorithm to generate core points, we are able to propose the following Magnanti-Wong algorithm for the FCJSP :

Algorithm 4.2 (Benders algorithm)

$\mathcal{P} \longleftarrow \emptyset, \quad LB \longleftarrow 0$

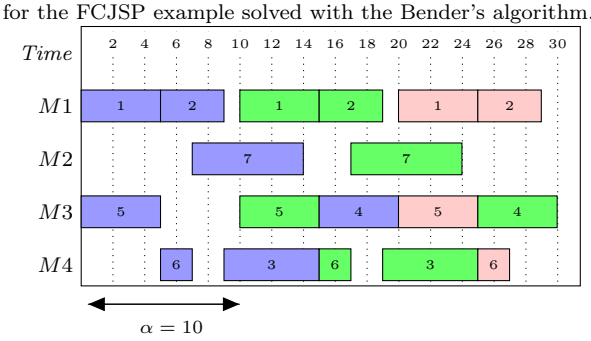
Step 0. Solve the master problem (7).
if infeasible then return infeasible.

Step 1. Solve the dual sub-problem (10). Obtain z^*.

Step 2. Find a core point y^0_k.

Step 3 Solve the Magnanti-Wong problem using y^0_k and z^*.
$\mathcal{P} \longleftarrow \mathcal{P} \cup \{\mathbf{u}^*\}, \quad UB \longleftarrow \overline{z}, \quad LB \longleftarrow \max\{LB, z^*\}$.
if $UB - LB > \epsilon$ then go to 0.
return current solution.

Fig. 7 Solution for the FCJSP example solved with the Bender's algorithm.



Where $\overline{z}$ denotes the current master problem objective value, and $\mathbf{u}^*$ denotes the solution of the Magnanti-Wong dual sub-problem.

Example 4.2 : We have solved Example 4 using our Benders decomposition algorithm. We have obtained the optimal cycle time $\alpha = 10$, with a task to machine assignment different from the one obtained in Example 4.1. The solution is displayed in Fig.7 as a Gantt diagram.

4.3 Heuristic procedure

The FCJSP is very hard to solve because it is highly combinatorial. For large instances, the solving time of the MILP presented in Section 5 can be very long. To tackle this issue, we have developed a heuristic procedure. Our approach consists in reducing the number of machines able to perform each task, i.e. the number of machines in the set $R(i), \forall i \in \mathcal{T}$. Let us define $R^H(i) \subset R(i)$ the reduced set of machines on which task $i \in \mathcal{T}$ can be assigned in the heuristic procedure. We call this approach "q-Flex", where $q = card(R^H(i))$. We solve linear program (12a-12d) to build $R^H(i)$ such that the total processing times per machine are equally balanced among all machines.

$$\min P_{max}$$

s.t.

$$P_{max} \geq \sum_{i \in \mathcal{T}} m_{i,r} p_{i,r}, \quad \forall r \in R \tag{12a}$$

$$\sum_{r \in r(i)} m_{i,r} = q, \quad \forall i \in \mathcal{T} \tag{12b}$$

$$P_{max} \in \mathbb{R} \tag{12c}$$

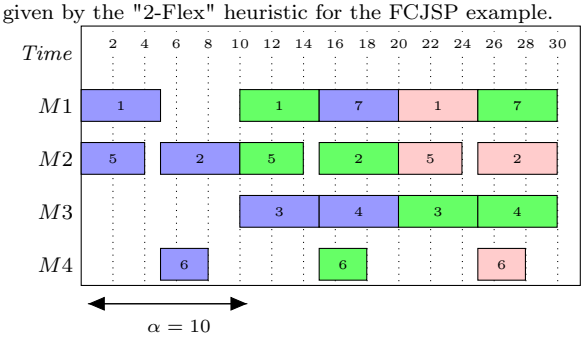
$$m_{i,r} \in \{0,1\}, \quad \forall i \in \mathcal{T}, \forall r \in R(i). \tag{12d}$$

For the heuristic to effectively reduce the flexibility of the problem, we must set q such that $q < card(R(i))$. We suggest to use $q = 2$ to guarantee an effective reduction of the solving time while keeping some flexibility in the problem but notice that this

heuristic is adjustable and a greater value of q could lead to better performance at the cost of a larger computational time.

Example 4.3 : We have solved Example 4 using this approach. We have obtained a cycle time $\alpha = 10$, with a task to machine assignment different from the one obtained in Example 4.1. The solution is displayed in Fig. 8 as a Gantt diagram.

Fig. 8 Solution given by the "2-Flex" heuristic for the FCJSP example.



5 Numerical experimentation

The following numerical experiments have been conduced on randomly generated instances with 10 tasks and 3, 4, or 5 machines. In these instances, each tasks can be assigned to any of the 3 to 5 machines, with an execution time between 10 and 19. For each kind of instance, we have generated 10 problems. To solve the MILP, we used the CLEX 12.7 solver. The computer we used is equipped with a dual-core i5-4210H processor and 8GB RAM and is running Ubuntu 18.04. We fixed the time limit to 3600 seconds. Note that the gap is defined as $\frac{\text{UB}-\tau^i}{\tau^i}$, where UB is the best upper bound found by the solver and τ^i is the value of τ in the considered solution.

In the following, we set $q = 2$ for the number of machines available for each task in the "q-Flex" heuristic procedure proposed in Section 4.3. Figure 9 displays the gaps between the solution found by the "2-Flex" heuristic procedure and the optimal solution found by the MILP and/or the Benders decomposition algorithm. These gaps are very measured and do not exceed 10% in average for all type of instances.

As we can see in Table 4, the MILP solving is very efficient for the easiest instances with 10 tasks and 5 machines. For those easy instances, it is much more efficient than the Benders decomposition, and there is no benefit to use the heuristic procedure since the execution times of both the MILP and the heuristic are very close and very fast, but the MILP gives better solutions. For instances of average difficulty with 10 tasks and 4 machines, the MILP is still more efficient than the Benders decomposition, but we can remark that the execution time of the MILP is increasing much faster than the execution time of the Benders decomposition. For these instances, the heuristic procedure generates solutions whose throughput is close to the optimal throughput with execution times that are much faster than the MILP's execution times. In short,

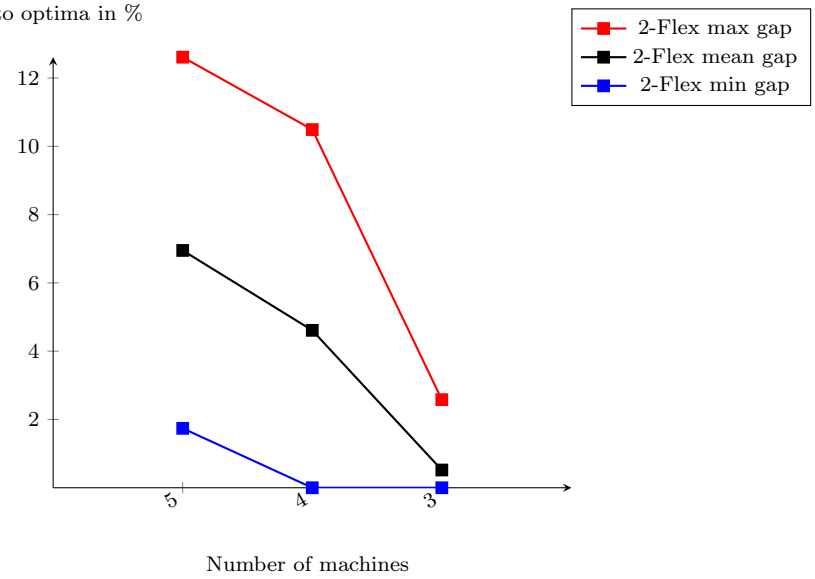
Table 4 Comparison of the solving times in seconds for the different approaches proposed in the paper. If the MILP times out, we have indicated the gap from the incumbent to CPLEX's best bound in parenthesis. Along with the heuristic's execution time, we displayed the gap to optima in parenthesis. Instances where the heuristic found the optimal throughput display "(-)" instead of the gap to optima value.

Instance name	MILP	Bender's decomposition	2-Flex heuristic
inst0_10tasks_5machines	0.76	292.75	0.27(4.57%)
inst1_10tasks_5machines	1.11	459.37	0.48(9.28%)
inst2_10tasks_5machines	0.15	342.91	0.52(12.61%)
inst3_10tasks_5machines	2.98	455.63	0.53(9.01%)
inst4_10tasks_5machines	0.44	336.79	0.17(3.35%)
inst5_10tasks_5machines	0.14	353.07	0.15(1.74%)
inst6_10tasks_5machines	5.78	353.23	0.46(10.21%)
inst7_10tasks_5machines	2.38	451.89	0.10(1.86%)
inst8_10tasks_5machines	0.83	470.23	0.20(10.37%)
inst9_10tasks_5machines	0.91	332.39	0.44(6.55%)
inst0_10tasks_4machines	29.37	505.57	1.78(10.49%)
inst1_10tasks_4machines	461.93	668.82	0.63(-)
inst2_10tasks_4machines	52.68	356.89	0.91(8.25%)
inst3_10tasks_4machines	132.53	703.93	1.04(2.53%)
inst4_10tasks_4machines	93.66	458.89	1.01(1.40%)
inst5_10tasks_4machines	19.78	364.25	0.43(5.44%)
inst6_10tasks_4machines	9.15	514.52	0.66(9.57%)
inst7_10tasks_4machines	49.81	455.74	0.31(2.97%)
inst8_10tasks_4machines	124.83	659.4	0.86(2.46%)
inst9_10tasks_4machines	14.65	317.62	0.87(2.97%)
inst0_10tasks_3machines	timeout	819.95	66.29(-)
inst1_10tasks_3machines	946.08	614.28	2.93(-)
inst2_10tasks_3machines	1640.75	516.42	6.74(-)
inst3_10tasks_3machines	823.53	589.63	27.07(-)
inst4_10tasks_3machines	658.47	385.83	34.43(2.58%)
inst5_10tasks_3machines	765.6	424.89	21.01(2.58%)
inst6_10tasks_3machines	timeout	954.92	7.62(-)
inst7_10tasks_3machines	2383.02	945.99	34.74(-)
inst8_10tasks_3machines	3215.08	769.91	9.81(-)
inst9_10tasks_3machines	2337.64	473.86	13.17(-)

for this type of instances, the heuristic is useful to quickly generate good feasible solutions. Finally, for the hard instances with 10 tasks and 3 machines, our Benders decomposition is always faster to find the optimal solution than the MILP. The heuristic procedure is also very efficient for this type of instance, often producing an optimal solution with fast execution times.

From these results, we learn that it is better to use the MILP for easy problems with numerous machines such as our instances with 10 tasks and 5 machines. For instances of average difficulty, there is a trade off between getting an optimal solution relatively slowly using the MILP or getting a close feasible solution very quickly with the heuristic procedure. However, for hard instances with a considerable number of disjunctions, such as the instances with 10 tasks and 3 machines, the execution times of the MILP rocket up and it is much better to use the heuristic procedure to obtain a feasible solution, or the Benders decomposition to obtain an optimal solution.

Fig. 9 Gap between the optimal solution found with the Benders decomposition algorithm and the solution generated by the heuristic procedure.



6 Conclusion

We have proposed a MILP modelling for the FCJSP where the objective is the minimisation of the cycle time. The problem is highly combinatorial, so we also proposed a Benders decomposition algorithm and a heuristic procedure that are more efficient for difficult instances. The Benders decomposition includes specific cuts to ensure the feasibility of the integer solution and uses the Magnanti-Wong algorithm to find Pareto-optimal optimality cuts. The heuristic reduces the number of machines available for each elementary task.

Numerical instances have shown that the MILP become inefficient for difficult instances with many disjunctions. The "q-Flex" heuristic procedure is a good way to tackle this issue since it is much faster than the MILP for these instances while producing solutions very close to the optimum. Also, if an optimal solution is required, our Benders decomposition is more efficient than the MILP for this type of instances.

References

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische mathematik 4(1):238–252

Bożejko W, Gnatowski A, Klempous R, Affenzeller M, Beham A (2016) Cyclic scheduling of a robotic cell. In: Cognitive Infocommunications (CogInfoCom), 2016 7th IEEE International Conference on, IEEE, pp 000379–000384

Bożejko W, Pempera J, Wodecki M (2017) A fine-grained parallel algorithm for the cyclic flexible job shop problem. Archives of Control Sciences 27(2):169–181

Brucker P, Kampmeyer T (2008a) Cyclic job shop scheduling problems with blocking. Annals of Operations Research 159(1):161–181
Brucker P, Kampmeyer T (2008b) A general model for cyclic machine scheduling problems. Discrete Applied Mathematics 156(13):2561–2572
Brucker P, Burke EK, Groenemeyer S (2012) A mixed integer programming model for the cyclic job-shop problem with transportation. Discrete applied mathematics 160(13-14):1924–1935
Błażewicz J, Domschke W, Pesch E (1996) The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93(1):1 – 33
Carlier J, Chrétienne P (1988) Problèmes d'ordonnancement: modélisation, complexité, algorithmes. Masson
Chretienne P, et al. (1991) The basic cyclic scheduling problem with deadlines. Discrete Applied Mathematics 30(2-3):109–123
Dell'Amico M, Trubian M (1993) Applying tabu search to the job-shop scheduling problem. Annals of Operations research 41(3):231–252
Draper DL, Jonsson AK, Clements DP, Joslin DE (1999) Cyclic scheduling. In: IJCAI, Citeseer, pp 1016–1021
Elmi A, Topaloglu S (2017) Cyclic job shop robotic cell scheduling problem: Ant colony optimization. Computers & Industrial Engineering 111:417–432
Fink M, Rahhou TB, Houssin L (2012) A new procedure for the cyclic job shop problem. IFAC Proceedings Volumes 45(6):69–74
Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research 35(9):2892–2907
Garey M, Johnson D (1979) Computers and intractability. W II Freeman and Company, New York
Gomes M, Barbosa-Povoa A, Novais A (2005) Optimal scheduling for flexible job shop operation. International journal of production research 43(11):2323–2353
Hamaz I, Houssin L, Cafieri S (2018a) A robust basic cyclic scheduling problem. EURO Journal on Computational Optimization 6(3):291–313
Hamaz I, Houssin L, Cafieri S (2018b) The Cyclic Job Shop Problem with uncertain processing times. In: 16th International Conference on Project Management and Scheduling (PMS 2018), Rome, Italy
Hanen C (1994) Study of a np-hard cyclic scheduling problem: The recurrent job-shop. European journal of operational research 72(1):82–101
Hanen C, Munier A (1995) A study of the cyclic scheduling problem on parallel processors. Discrete Applied Mathematics 57(2-3):167–192
Hillion HP, Proth JM (1989) Performance evaluation of job-shop systems using timed event-graphs. IEEE transactions on automatic control 34(1):3–9
Houssin L (2011) Cyclic jobshop problem and (max, plus) algebra. In: World IFAC Congress (IFAC 2011), pp 2717–2721
Ioachim I, Soumis F (1995) Schedule efficiency in a robotic production cell. International Journal of Flexible Manufacturing Systems 7(1):5–26
Jalilvand-Nejad A, Fattahi P (2015) A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. Journal of Intelligent Manufacturing 26(6):1085–1098
Jamili A, Shafia MA, Tavakkoli-Moghaddam R (2011) A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling

problem. The International Journal of Advanced Manufacturing Technology 54(1-4):309–322
Kampmeyer T (2006) Cyclic scheduling problems
Karp RM, Orlin JB (1981) Parametric shortest path algorithms with an application to cyclic staffing. Discrete Applied Mathematics 3(1):37–45
Kats V, Levner E (1996) Polynomial algorithms for scheduling of robots. Intelligent Scheduling of Robots and FMS pp 77–100
Kim HJ, Lee JH (2018) Cyclic robot scheduling for 3d printer-based flexible assembly systems. Annals of Operations Research pp 1–21
Korbaa O, Camus H, Gentina JC (2002) A new cyclic scheduling algorithm for flexible manufacturing systems. International Journal of Flexible Manufacturing Systems 14(2):173–187
Levner E, Kats V (1998) A parametric critical path problem and an application for cyclic scheduling. Discrete Applied Mathematics 87(1-3):149–158
Magnanti TL, Wong RT (1981) Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. Operations research 29(3):464–484
Papadakos N (2008) Practical enhancements to the magnanti–wong method. Operations Research Letters 36(4):444–449
Quinton F, Hamaz I, Houssin L (2018) Algorithms for the flexible cyclic jobshop problem. In: 14th IEEE International Conference on Automation Science and Engineering, CASE 2018, Munich, Germany, August 20-24, 2018, pp 945–950
Rossi A, Dini G (2007) Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. Robotics and Computer-Integrated Manufacturing 23(5):503–516
Roundy R (1992) Cyclic schedules for job shops with identical jobs. Mathematics of operations research 17(4):842–865
Saidi-Mehrabad M, Fattahi P (2007) Flexible job shop scheduling with tabu search algorithms. The International Journal of Advanced Manufacturing Technology 32(5-6):563–570
Schutten JM (1998) Practical job shop scheduling. Annals of Operations Research 83:161–178
Serafini P, Ukovich W (1989) A mathematical model for periodic scheduling problems. SIAM Journal on Discrete Mathematics 2(4):550–581
Song JS, Lee TE (1998) Petri net modeling and scheduling for cyclic job shops with blocking. Computers & Industrial Engineering 34(2):281–295
Thomalla CS (2001) Job shop scheduling with alternative process plans. International Journal of Production Economics 74(1):125 – 134
Xia W, Wu Z (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering 48(2):409–425
Zhang H, Collart-Dutilleul S, Mesghouni K (2015) Cyclic scheduling of flexible job-shop with time window constraints and resource capacity constraints. IFAC-PapersOnLine 48(3):816 – 821