Algorithms for the Flexible Cyclic Jobshop Problem

Félix Quinton, Idir Hamaz, Laurent Houssin

Algorithms for the flexible cyclic jobshop problem

Félix Quinton, Idir Hamaz[1], Laurent Houssin[1]

***Abstract*— This paper considers the cyclic jobshop problem in a flexible context. The aim is to find the minimum cycle time of a periodic schedule. The flexibility feature comes from the ability of the machines or robots to perform several kinds of tasks. Hence, the scheduling problem does not only concern starting time of tasks but also on which machines the tasks will be performed. We propose an exact method to solve this problem and two heuristics.**

I. INTRODUCTION

In automation systems, a robotic cell for instance, actions are mainly cyclic. Once an occurrence of each task is planed and the cycle time is determined, the schedule is completely defined. Each occurrence is then performed periodically. Numerous papers consider the scheduling problem of a robotic cells in a cyclic manner (Crama [4], Yan [5], Bożejko [6]).

In scheduling theory, cyclic jobshop scheduling and flexible jobshop scheduling are long studied NP-hard problems (Hanen [1], Brucker [2], Levner [8]) apart from each other.

However few papers tackle the flexible cyclic scheduling problem despite the capabilities of robots to perform several types of task. Because of the highly combinatorial nature of flexible scheduling problems (Xia [9]), research focused on genetic algorithm to solve the flexible cyclic scheduling problem (Jalilvand [10], Bożejko [11]).

This paper is organized as follows : in Section II. we describe the basic cyclic scheduling problem. Section III. describes the cyclic jobshop scheduling problem in which we introduce the notion of robots and resource constraint. In Section IV. we introduce the flexible cyclic jobshop problem and the notion of flexibility which means that a robot can perform several different tasks, and propose a mathematical model for this problem. In Section V. we propose two heuristic procedures to tackle long solving time for large instances. Finally, we present numerical instances in Section VI.

II. Basic cyclic scheduling problem

The basic cyclic scheduling problem (BCSP) is an extension of the basic scheduling problem where a schedule of elementary tasks $i \in \mathcal{T} = \{1, ..., n\}$ is infinitely repeated. For each task $i \in \mathcal{T}$, we are given its execution time p_i and we denote by $t_i(k)$ the starting time of occurrence k of task $i \in \mathcal{T}$.

I. Hamaz and L. Houssin are with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France laurent.houssin@laas.fr

Elementary tasks are connected by precedence constraints. A precedence constraint (i, j, p_i, H_{ij}), where H_{ij} is called the height of the precedence constraint and represents the occurrence shift between tasks i and j, state that :

$$t_j(k + H_{ij}) \geq p_i + t_i(k)$$

Let us denote $t_i = t_i(0), i \in \mathcal{T}$. In this study, we are interested in periodic schedules, which are such that there exists a cycle time α satisfying $t_i(k) = \alpha k + t_i$.

The objective considered in this study is the minimization of the cycle time α which is the time elapsed between two occurrences of the task.

A BCSP can be represented on a graph where the set of edges $\mathcal{E}$ represent the precedence constraints and vertices represent elementary tasks. The edges are weighted and the weight of a given edge represent the execution time of the task at its origin and the height of the precedence constraint.

There are two methods to solve the BCSP. The first is based on graph and consist to find the maximum mean cycle in the graph.

The second is to use mathematical programming. The BCSP can be expressed as a linear programming problem as follows.

$$\min \alpha$$

s.t.

$$\alpha \geq p_i, \quad \forall i \in \mathcal{T} \tag{1a}$$

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

$$t_i \geq 0, \quad \forall i \in \mathcal{T} \tag{1c}$$

Constraints (1a) correspond to non-reentrance constraints which ensure that two occurrences of the same elementary task do not overlap. Constraints (1b) set the precedence constraints.

Example 1: Table I reports the data of a BCSP instance with 7 elementary tasks. The graph representing this example is presented in Fig. 2, where the red dotted edges are ignored. After solving this BCSP, we find an optimal cycle time of 9.

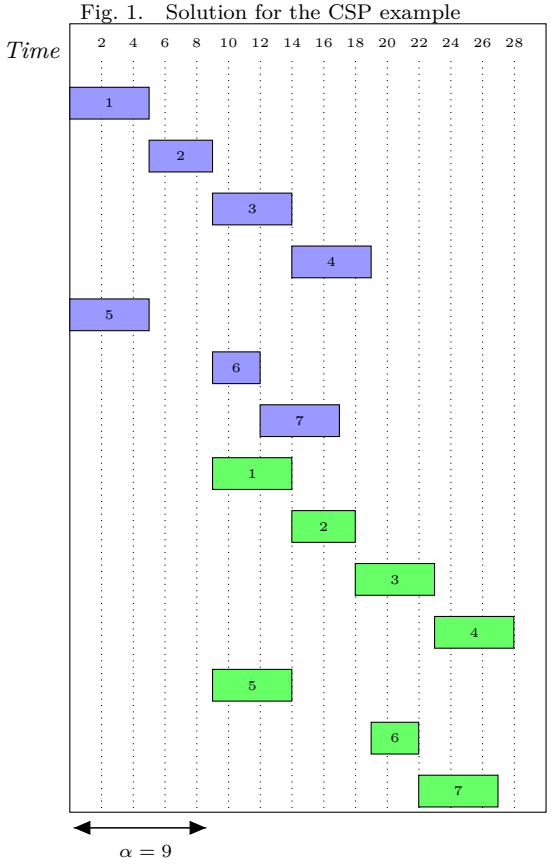
TABLE I

An exemple for the BCSP

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

The solution for the BCSP instance is given in Fig. 1 in the form of a Gantt diagram :

Fig. 1. Solution for the CSP example



III. Cyclic jobshop scheduling problem

In the cyclic jobshop scheduling problem (CJSP), each elementary task $i \in \mathcal{T}$ is assigned to a robot $r_i \in \mathcal{R} = \{1, ..., R\}$, with $R < n$.

In consequence, the lack of resource induces some disjunction constraints. For a pair of tasks $(i, j) \in \mathcal{T}^2, i \neq j$ such that $r_i = r_j$, an occurrence of tasks i and an occurrence of j, cannot be executed at the same time. Denote by $D = \{(i, j) | R(i) \cap R(j) \neq \emptyset\}$ the set of pair of tasks in disjunction.

More precisely, this disjunction leads to a pair of constraints :

$$\begin{cases} t_j(k + K_{ij}) \geq t_i(k) + p_i \\ t_i(k + K_{ji}) \geq t_j(k) + p_j \end{cases}$$

Where K_{ij} (resp. K_{ji}) is an occurrence shift between task i and j (resp. j and i), and $K_{ij} + K_{ji} = 1$ as showed by Hanen [1]. Note that in this problem the variables are α, $(t)_{i \in \mathcal{T}}$, and $(K)_{(i,j) \in D}$.

A job is composed by elementary tasks and precedence constraints between these tasks.

A feature of the CJSP is the Work In Process (WIP). It corresponds to the maximum number of occurrences of a job processed simultaneously.

The CJSP can be resolved using a Branch-and-Bound procedure (Houssin [8], Hanen[1]) or using mathematical programming. The CJSP can be represented as a Mixed Interger Non Linear Programming problem (MINLP) as follows :

$$\min \alpha$$

s.t.

$$\alpha \geq p_i, \quad \forall i \in \mathcal{T} \tag{2a}$$
$$t_j + \alpha H_{i,j} \geq t_i + p_i, \quad \forall (i, j) \in E \tag{2b}$$
$$t_j + \alpha K(i, j) \geq t_i + p_i, \quad \forall (i, j) \in D \tag{2c}$$
$$K(i, j) + K(j, i) = 1, \quad \forall (i, j) \in D \tag{2d}$$
$$K(i, j) \in \mathbb{Z}, \quad \forall (i, j) \in D \tag{2e}$$
$$t_i \geq 0, \quad \forall i \in \mathcal{T} \tag{2f}$$

Constraints (2a) are the non-reentrance constraints and constraints (2b) are the precedence constraints. Constraints (2c) are the disjunction constraints defined above, and constraints (2d) state for the usual CJSP property previously enunciated.

However, non linear programming is very unpractical. Consequently, we want to linearize the model. Hanen [1] 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 the CJSP can be written as follows (Hanen [1]):

$$\max \tau$$

s.t.

$$\tau \leq \frac{1}{p_i}, \quad \forall i \in \mathcal{T} \tag{3a}$$
$$u_j + H_{i,j} \geq u_i + \tau p_i, \quad \forall (i, j) \in E \tag{3b}$$
$$u_j + K(i, j) \geq u_i + \tau p_i, \quad \forall (i, j) \in D \tag{3c}$$
$$K(i, j) + K(j, i) = 1, \quad \forall (i, j) \in D \tag{3d}$$
$$K(i, j) \in \mathbb{Z}, \quad \forall (i, j) \in D \tag{3e}$$
$$u_i \geq 0, \quad \forall i \in \mathcal{T} \tag{3f}$$

Example 2: in Table II, we have updated Example 1 so that it fits the CJSP with 4 robots. The graph in Fig. 2 shows the problem with only the disjunctive constraints for robot M_3 displayed for cleanliness purposes. Solving this CJSP, we find an optimal cycle time $\alpha = 13$.

TABLE II
An exemple 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

The solution for this CJSP is given in Fig.3 in the form of a Gantt diagram :

IV. Flexible jobshop cyclic scheduling problem

The flexible cyclic jobshop scheduling problem (FCJSP) is a CJSP where the elementary tasks are

Fig. 2. Associated graph to the CJSP instance described in example 2

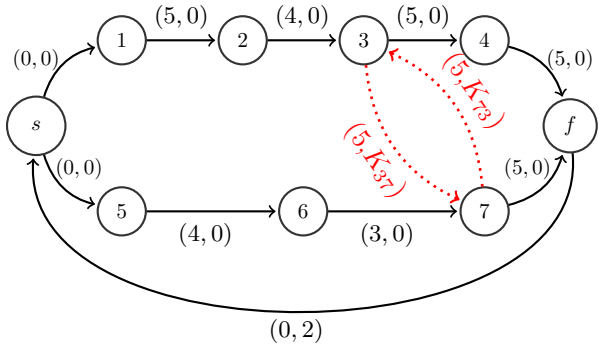
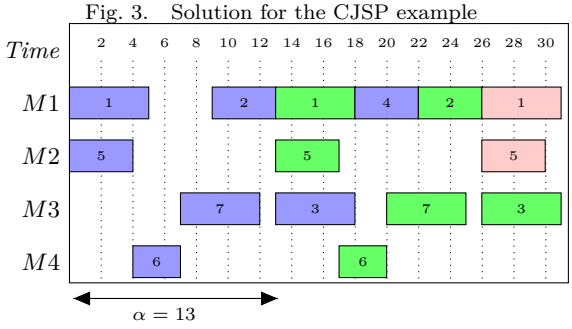


Fig. 3. Solution for the CJSP example



flexible. This means that for each task $i \in \mathcal{T}$, we have a set $R(i) \subset \mathcal{R}$ of robots on which task i can be assigned.

In consequence, the assignment of a task i to a robot $r \in R(i)$ is a decision variable. Each assignment of a task $i \in \mathcal{T}$ to a robot $r \in R(i)$ is associated with a given execution time p_{ir}.

Also, because we do not know *a priori* on which robot each task will be assigned, we do not know the set D of pairs of tasks which are connected by a disjunctive constraint.

Example 3: in Table III, we have updated Example 2 so that it fits the FCJSP. Each task can be assigned to a subset of 3 out of 4 robots. Note that the new possibilities come with a higher execution time.

A. Linear model

In this section, we propose a mixed integer linear programming (MILP) formulation for the FCJSP. Let $R(i,j) = R(i) \cap R(j), \forall i,j \in \mathcal{T}$ the robots available both for tasks i and j. Recall that E is the set of precedence constraints and that D is the set of disjunctive constraints.

We also define some decision variables for the assignation of a task $i \in \mathcal{T}$ to a robot $r \in R(i)$:

$$\forall i = \{1, ..., n\}, \forall r \in R(i),$$
$$m_{i,r} = \begin{cases} 1 \text{ if task i is assigned to robot r} \\ 0 \text{ sinon} \end{cases}$$

TABLE III
AN EXEMPLE FOR THE FCJSP

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

The MILP for the FCJSP can be formulated as :

$$\max \tau$$

s.t.

$$\tau \leq \frac{1}{p_{i,r}} + (1 - m_{i,r})P_1, \quad \forall i \in T, \forall r \in R(i) \tag{4}$$

$$u_j + H_{i,j} \geq u_i + p_{i,r}(\tau - (1 - m_{i,r})), \quad \forall (i,j) \in E, \forall r \in R(i) \tag{5}$$

$$u_j + K_{i,j} \geq u_i + p_{i,r}(\tau - (2 - m_{i,r} - m_{j,r})), \quad \forall (i,j) \in E, \forall r \in R(i,j) \tag{6}$$

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

$$K(i,j) + K(j,i) = 1, \quad \forall (i,j) \in D \tag{8}$$

$$K(i,j) \in \mathbb{Z}, \quad \forall (i,j) \in D \tag{9}$$

$$u_i \geq 0, \forall i \in T \tag{10}$$

$$m_{i,r} \in \{0,1\}, \quad \forall i \in T \forall r \in R(i) \tag{11}$$

Constraints (4) correspond to non-reentrance constraints. They ensure that two occurrences of the same elementary task do not overlap. The P_1 term allows us to deactivate the constraints for which task i is not affected to robot r. Let us set $P_1 = \frac{1}{\min\limits_{i \in \mathcal{T}, r \in R(i)} p_{i,r}}$.

Constraints (5) set the precedence constraints, i.e. the order in which tasks have to be executed. Those constraints must be activated only if task i is assigned to robot r, because we use the term $p_{i,r}$. All constraints in (5) such that $m_{i,r} = 0$ are dominated by the constraints in (5) such that $m_{i,r} = 1$ because $p_{i,r}(\tau - 1) < 0 < p_{i,r}\tau$, and thus are deactivated.

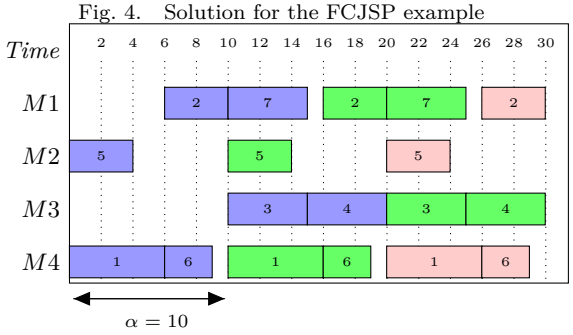
Constraints (6) represents the disjunctive constraints, ensuring that two tasks are not executed at the same time period on the same robot. We want those constraints to be activated only when tasks i and j are assigned to the same robot r. Let $r \in R(i,j)$ a robot on which both tasks i and j can be executed, then if task i or task j is not assigned to r, we have $\tau - (2 - m_{i,r} - m_{j,r}) < 0$, which deactivate the corresponding disjunctive constraint. By

the same mechanism as previously, constraints where $m_{i,r} = 0$ or $m_{j,r} = 0$ are dominated by those where $m_{i,r} = m_{j,r} = 1$ and thus are deactivated.

Constraints (7) ensure that every task $i \in \mathcal{T}$ is assigned to a robot $r \in R(i)$. Constraints (8) are usual constraints for the CJSP, introduced by Hanen [1].

Example 3.1: We have solved the Example 3. with our MILP. We have obtained a cycle time $\alpha = 10$. The solution is displayed as Gantt diagram in Fig. 4.

Fig. 4. Solution for the FCJSP example



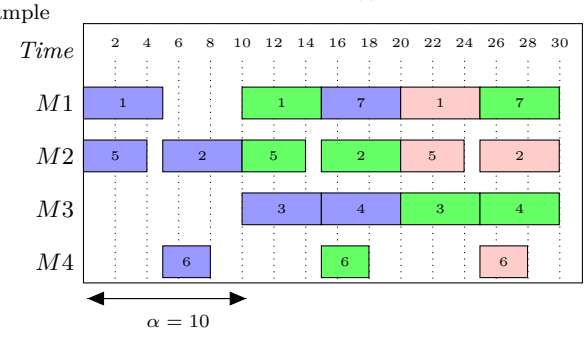
B. Heuristic procedures

The FCJSP is a difficult problem because it is highly combinatorial. The solving time of the MILP presented in V. can become very long for large numerical instances. To tackle this issue, we have developed two heuristic procedures.

1) Reducing the $R(i)$ sets: a first approach is to reduce the flexibility of the problem, i.e. the number of robots in the set $R(i), \forall i \in \mathcal{T}$. Define $R^H(i) \subset R(i)$ the set of robot on which task $i \in \mathcal{T}$ can be assigned in the heuristic procedure. The set $R^H(i)$ can be fixed as the set of R^H robots with the lowest execution time for task i, with $R^H < R$. In our numerical experimentation, we have set $R^H = 2$, which allows us to obtain some good feasible solutions in a reasonable solving time without losing flexibility.

Example 3.2: we have solved Example 3 using this approach. We have obtained a cycle time $\alpha = 10$, with a different task to machine assignment. The solution is displayed in Fig. 5 as a Gantt diagram.

Fig. 5. Solution given by the "reduce $R(i)$" heuristic for the FCJSP example



2) Critical circuit heuristic: an other approach is to allow only a fraction of the tasks to be flexible. In other words, we chose a subset $I \subset \mathcal{T}$ of tasks whose assignment to robot will be a decision variable while other tasks $i \in T \backslash I$ are already assigned.

We define I as the set of tasks composing the critical circuit of our FCJSP. The critical circuit in an oriented bi-valuated graph is the circuit minimizing $\frac{H_c}{L_c}$ (Hanen [1]), where L_c is the sum of the edge's length of the circuit and H_c is the sum of the edge's height of the circuit.

The heuristic procedure consists in iteratively solving the reduced problem where only the tasks composing the critical circuit are flexible until finding twice the same critical circuit. It can be written as follows :

Step 0. Initialise by solving the CJSP generated by the previous heuristic with $R^H = 1$.

Step 1. Find the critical circuit using the Howard algorithm [3].

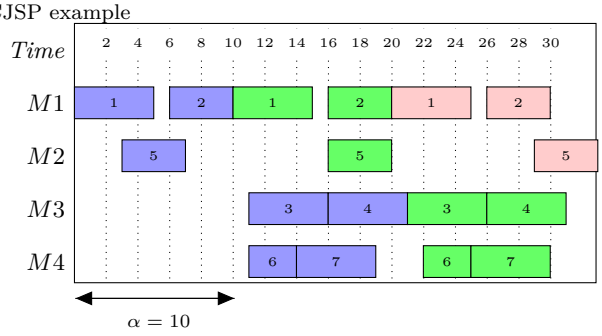
Step 2. Solve the problem allowing only the tasks composing the critical circuit to be flexible.

Step 2.1 Find a critical circuit.

Step 2.2 If the critical circuit has not been explored already, go to 2.

3) Example 3.3: we have solved Example 3 using the critical circuit procedure. Once again, we obtained a cycle time $\alpha = 10$ with a different task to machine assignment. The solution is displayed in Fig. 6. in the form of a Gantt diagramm.

Fig. 6. Solution given by the "critical circuit" heuristic for the FCJSP example



V. Numerical experimentation

The following numerical experiments has been conduced on randomly generated instances with 15, 20 and 30 tasks and 5 robots. Also, each tasks can be assigned to 3 to 5 of the robots. We fixed the time limit to 300 seconds for the MILP solving and 60 seconds for the heuristic procedures. The mean times displayed below does not account for the instances whose resolution time exceed the time limit, and the mean gaps does not account for the instances that have been solved before the time limit. 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 incumbent solution.

As we can see in Table IV, the MILP is unable to solve instances larger than 20 tasks and 4 assignment possibilities for each task.

Also, we can see through Tables V. to VII. that the "reducing $R(i)$ sets" heuristic procedure proposed in IV.B.1. with $R^H = 2$ is more efficient than the "critical circuit" heuristic procedure proposed in IV.B.2. on every aspect.

On very large instances, we can see in Table VI. that the "reducing $R(i)$ sets" heuristic procedure is more efficient than the MILP.

TABLE IV
NUMERICAL RESULTS WITH THE MILP

Instance parameters	Solving time (sec)	Gap to optima	Number of solved instances
15 tasks | 3 robots	25.09	undefined	10
15 tasks | 4 robots	36.58	11.11%	9
15 tasks | 5 robots	5.13	16.67%	9
20 tasks | 3 robots	111.87	17.29%	8
20 tasks | 4 robots	115.25	13.06%	8
20 tasks | 5 robots	time out	23.81%	0
30 tasks | 3 robots	time out	32.64%	0
30 tasks | 4 robots	time out	123.61%	0
30 tasks | 5 robots	time out	121.95%	0

TABLE V
COMPARISON OF THE MEAN SOLVING TIME FOR THE MILP WITH THE HEURISTIC PROCEDURES

Instance parameters	MILP	Reduce $R(i)$ heuristic	Critical circuit heuristic
15 tasks | 3 robots	25.09	3.39	7.05
15 tasks | 4 robots	36.58	5.80	9.56
15 tasks | 5 robots	5.13	0.86	3.20
20 tasks | 3 robots	111.87	54.25	26.51
20 tasks | 4 robots	115.25	27.02	32.01
20 tasks | 5 robots	time out	38.11	26.90
30 tasks | 3 robots	time out	60.0	60.0
30 tasks | 4 robots	time out	60.0	53.80
30 tasks | 5 robots	time out	60.0	48.83

TABLE VI
NUMBER OF INSTANCES WHERE THE HEURISTIC SOLUTION $\geq$ THE MILP INCUMBENT

Instance parameters	Reduce $R(i)$ heuristic	Critical circuit heuristic
15 tasks | 3 robots	7	3
15 tasks | 4 robots	4	5
15 tasks | 5 robots	10	5
20 tasks | 3 robots	5	2
20 tasks | 4 robots	9	3
20 tasks | 5 robots	10	7
30 tasks | 3 robots	3	3
30 tasks | 4 robots	10	6
30 tasks | 5 robots	10	9

TABLE VII
MEAN GAP BETWEEN THE HEURISTIC SOLUTION AND THE MILP INCUMBENT WHEN THE MILP INCUMBENT > THE HEURISTIC SOLUTION

Instance parameters	Reduce $R(i)$ heuristic	Critical circuit heuristic
15 tasks | 3 robots	4.33%	11.39%
15 tasks | 4 robots	7.05%	15.70%
15 tasks | 5 robots	undefined	13.95%
20 tasks | 3 robots	4.69%	11.36%
20 tasks | 4 robots	10%	12.61%
20 tasks | 5 robots	undefined	22.11%
30 tasks | 3 robots	3.46%	11.11%
30 tasks | 4 robots	undefined	10.26%
30 tasks | 5 robots	undefined	32.52%

VI. CONCLUSION

We have proposed a MILP for the FCJSP and two heuristic procedures that are more efficient for large instances.

Further work could be to improve our heuristic procedures and to consider the possibility of a Bender's or Danzwig-Wolfe decomposition.

REFERENCES

[1] C.HANEN, "Study of a NP-hard cyclic scheduling problem: The recurrent jobshop", *European journal of operational research* Université P. et M. Curie, 1994.

[2] Peter Brucker, Sigrid Knust, "jobshop Problems with Flexible Machines", *Complex scheduling*, Springer, 2004.

[3] Stephane Gaubert, Max Plus, "Methods and applications of (max,+) linear algebra", *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, 1997.

[4] Crama, Yves, and Joris Van De Klundert, "Cyclic scheduling of identical parts in a robotic cell", *Operations Research 45.6* (1997): 952-965

[5] Yan, Pengyu, et al., "An algorithm for optimal cyclic scheduling in a robotic cell with flexible processing times", *Industrial Engineering and Engineering Management, 2008*, IEEM 2008, *IEEE International Conference on* (pp. 153-157)

[6] Bożejko, Wojciech, et al., "Cyclic scheduling of a robotic cell", *Cognitive Infocommunications (CogInfoCom), 2016 7th IEEE International Conference on.* IEEE, 2016.

[7] Fink M, Rahhou TB, Houssin L., "A new procedure for the cyclic job shop problem.", *IFAC Proceedings Volumes*, 2012

[8] Levner, Eugene, et al., "Complexity of cyclic scheduling problems: A state-of-the-art survey." *Computers & Industrial Engineering*, 2010

[9] Xia, W., & Wu, Z., An effective hybrid optimization approach for multi-objective flexible jobshop scheduling problems. *Computers & Industrial Engineering*, 2005

[10] Jalilvand-Nejad, Amir, and Parviz Fattahi. "A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem." *Journal of Intelligent Manufacturing*, 2015

[11] Bożejko, Wojciech, Jarosław Pempera, and Mieczysław Wodecki, "A fine-grained parallel algorithm for the cyclic flexible job shop problem.", *rchives of Control Sciences*, 2017