An exension of the heap of pieces model for the cyclic jobshop problem

Laurent Houssin

***Abstract*— This paper considers Cyclic Jobshop Problem, which aim to find the tasks processing order for each machine that maximizes the throughput or equivalently minimizes the cycle time. We develop a branch and bound enumeration procedure to solve the Cyclic Jobshop Problem, based on heap of pieces theory and $max,+$ algebra. We propose an extension of this model to consider more structures of $K-$cyclic schedules. It provides more realistic values for the evaluation of the cycle time and improve the efficiency of the approach.**

I. Introduction

In classical scheduling, a set of tasks is executed once while the determined schedule optimizes objective functions such as the makespan or earliness-tardiness. In contrast, cyclic scheduling means performing a set of generic tasks infinitely often while minimizing the time between two occurrences of the same task. Cyclic scheduling has several applications, e.g. in robotic industry (see [1]), in manufacturing systems (see [2] and [3]) or multiprocessor computing (see [4]). It has been studied from multiple perspectives, since several representations of the problem exists such as graph theory, mixed integer linear programming, Petri nets or $(max,+)$ algebra. An overview about cyclic scheduling problems and the different approaches can be found in [5].

For our concern we tackle the Cyclic Jobshop Scheduling Problem (CJSP). This problem is proven to be NP-hard by [6]. We employ graph theory to represent the problem (see [7] and [5]). The goal of the CJSP is to find a periodic schedule with minimal asymptotic cycle time that satisfies all uniform and disjunctive constraints. This problem can be derived in a mixed integer linear program (MILP) formulation decribed in [7] and [6]. However, a less known tool is the *heaps of pieces* approach. The seminal paper considering this modelling for cyclic scheduling is [8] in which the authors introduce a method to represent safe Petri nets as particular automata allowing the computation of the height of heaps of pieces. This approach enables to compute efficiently the throughput of a cyclic schedule by means of $(max,+)$ algebra. In this paper, we use the heaps of pieces approach for solving cyclic scheduling problems with resource constraints.

The paper is organized as follows. In section 2, we first introduce the CJSP and the parameters of the problem (work-in-process and cyclicity). The MILP formulation is also presented. Section 3 is devoted to the heap of pieces theory. We recall $(max,+)$ algeabric tools in this section. We propose an extension of the heap of pieces model and we design a branch-and-bound procedure in section 4. A simple numeric example is developed throughout the paper.

L. Houssin is 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

II. Cyclic scheduling problems

A. The Basic Cyclic Scheduling Problem

We first remind the Basic Cyclic Scheduling Problem (BCSP). This problem involves generic tasks and precedence constraints between tasks but no resource constraints are considered. A set $\mathscr{T}$ of *n generic tasks* is processed in parallel by an unbounded number of machines and there is a set of q precedence constraints. We denote this set $\mathscr{A} = \{a_1,\dots,a_q\}$ where each element corresponds to a constraint represented by a triple (i,j,h). More precisely, the *uniform constraint* (i,j,h) means that

$$s_i(k)+p_i \leq s_j(k+h),$$

where s_i denotes the beginning of the operation i, p_i is the processing time of i. The variable h is usually called the *height* of the constraint. In this framework, the asymptotic cycle time $\alpha(S)$ is usually minimized (with S a feasible schedule). Equivalently we can aim at maximizing the throughput $r(S)=\frac{1}{\alpha(S)}$.

A directed graph $G=(T,E)$ with a vertex set T and a set of arcs E can be associated with a BCSP such that a node (resp. an arc) of G corresponds to a task (resp. precedence constraints) in the BCSP. Each arc (i,j) of G is equipped with two values: the processing time p_i of the task i and the height (also called event shift) $H_{ij} \in \mathbb{Z}$.

Since this problem doesn't involve any resource problem, it can be easily solved with linear programming or $(max,+)$ algebra [5]. Moreover, it can be shown that the minimum cycle time of the system is given by the critical circuit of the graph. Several algorithms exist to compute the critical circuit. In the following, we briefly present three different ways to solve this problem. We can first mention the algorithm from [9] which runs in $O(n^3 log(n))$. Second, Karp's algorithm in [10], [11] can be used to find the critical circuit in $O(n^3)$. However, Karp's algorithm only works when for all existing h in (i,j,h), we have $h=1$. When $h>1$, a remodeling is necessary (see [12]). In cyclic scheduling, also negative heights can occur frequently. (*cf.* the basic example of [6]). A first attempt for considering negative heights in Karp's algorithm was made in [13]. Howard's algorithm ([14]), originally designed for Markov decision processes, is adapted to the framework of $(max,+)$ algebra in [15] and can be used to compute the critical circuit of a double weighted directed graph . Although the complexity remains unproved it shows excellent performance and almost linear running time.

We can distinguish two categories of cyclic schedules: 1-periodic schedules and K-periodic schedules. The first category is characterized by a period α such that :

$$s_i(k+1) = \alpha + s_i(k), \quad \forall i \in \mathscr{T}, \quad \forall k \geq 0.$$

Whereas K-periodic schedules are also defined by period which corresponds to a fixed interval time between any K consecutive occurences of i :

$$s_i(k+K) = \alpha_K + s_i(k), \quad \forall i \in \mathscr{T}, \quad \forall k \geq 0.$$

In this case, the asymptotic cycle time is then defined by $\frac{\alpha_K}{K}$.

B. The Cyclic Jobshop Scheduling Problem

For our concern, we are interested in the cyclic job shop problem. In this case, tasks are *a priori* mapped onto machines and the number of machines is smaller than the number of tasks to perform. More precisely, a cyclic job shop is defined by :

- a set $\mathscr{T}$ of elementary tasks,
- a set $\mathscr{R}$ of machines,
- for each task $t \in \mathscr{T}$, a processing time p_t and a machine $m_t \in \mathscr{R}$ on which the task has to be performed,
- a set $\mathscr{A}$ of uniform constraints (as defined above),
- a set of jobs $\mathscr{J}$ corresponding to a production sequence of elementary tasks. More precisely, a job J_1 defines a sequence $J_1 = t_{11} \dots t_{1k}$ to be executed in this order.

Regarding the cycle time evaluation, the problem is the same as in §II-A since it still is the search of maximum circuit ratio in a graph. The graph of the CJSP is slightly different from the BCSP since resource constraints add arcs in the graph of the BCSP (see [16] for details). These *disjunctive* arcs occurs between two tasks mapped on the same machine and an arc (i, j) is labelled with the processing time p_i and an event shift K_{ij} that describe if task i is performed before task j (see [16] for details).

Besides a mixed integer linear program is designed in [6] from the graph representation in order to solve the problem. Remodeling is necessary because the resource constraint features a non-linear component with $\alpha \times K_{ij}$. Thus, the throughput $\tau = \alpha^{-1}$ is introduced together with a new variable $u_i = s_i(0) \times \alpha^{-1}$, which leads to a linear formulation that can be solved with a linear programming solver.

$$\max \tau \tag{1}$$

$$\text{s.t.}$$

$$u_j \geq 0 \ , \ u_i \geq 0 \tag{2}$$

$$u_j - u_i \geq \tau \times p_i \qquad i, j \in \mathscr{T} \tag{3}$$

$$u_j - u_i \geq \tau \times p_i - K_{ij} \qquad i, j \in \mathscr{T} \tag{4}$$

$$K_{ij} + K_{ji} = 1 \qquad K_{ij}, K_{ji} \in \mathbb{Z}, \ \forall i, j \in T_s, \ s \in \mathscr{M} \tag{5}$$

In this problem, two parameters define the structure of the schedule. The first one is the Work-In-Process (WIP), it corresponds to the maximum number of occurrences of job that can be simultenaously processed. Let us consider the example 1. The figure 1 shows a schedule (a solution of example 1) with a WIP=1 and the figure 2 shows the same schedule with a WIP=2. We can notice that at time $t = 7$, 2 occurrences of the Job J_1 are in process in the schedule of Fig. 2 whilst this scheduling structure is not allowed if WIP=1.

Example 1:

Job	J_1			J_2		J_3		J_4	
Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Processing time	1	3	3	1	2	2	1	2	1
Machine	M_1	M_2	M_3	M_3	M_2	M_1	M_3	M_1	M_3

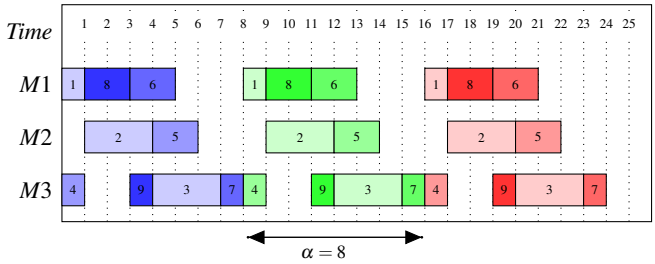

Fig. 1. 1-cyclic schedule with WIP=1

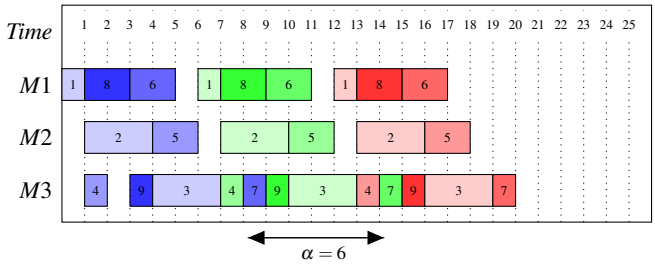

Fig. 2. 1-cyclic schedule with WIP=2

The second parameters has already been discussed in the section II-A. Indeed, the structure of the schedule is highly dependant of the cyclicity. Previous studies of this problem have shown that K-periodic schedules are dominant ([5]) and the problem is NP-hard ([6]) for throughput maximization. Figure 3 illustrates an example of K-cyclic solution of the problem in example 1.

III. The Heap of Pieces Model

We first recall some algebraic tools concerning $(max, +)$ algebra. The $(max, +)$ semiring is the set $\mathbb{R} \cup \{-\infty\}$ endowed with the max operator, written $a \oplus b = max(a, b)$, and the usual sum written $a \otimes b = a + b$. The sum (resp. product) admits a neutral element denoted $\varepsilon = -\infty$ (resp. $e = 0$), it leads to $a \oplus \varepsilon = a$ and $a \otimes e = a$. For matrices, additions and products give $(A \oplus B)_{ij} = A_{ij} \oplus B_{ij}$ and $(A \otimes B)_{ij} = \bigoplus_{k=1}^{n} A_{ik} \otimes B_{kj}$.

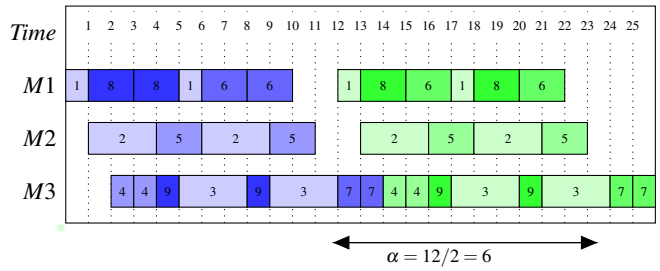

Fig. 3. 2-cyclic schedule with WIP=2

An exhaustive presentation of $(max,+)$ algebra can be found in [12]. We now present the heap model structure that was introduced in [8].

Definition 1 (Heap model): A heap model is composed by :

- $\mathscr{P}$ a finite set of pieces,
- $\mathscr{S}$ a finite set of slots,
- R gives the subset of slots occupied by a piece,
- $l : \mathscr{P} \times \mathscr{S} \to \mathbb{R} \cup \{-\infty\}$ gives the lower contour of a piece,
- $u : \mathscr{P} \times \mathscr{S} \to \mathbb{R} \cup \{-\infty\}$ gives the upper contour of a piece.

For each piece p (possibly non connected) of a heap model, we define the matrix $\mathscr{M}(p)$ of dimension $|\mathscr{S}| \times |\mathscr{S}|$ by

$$\mathscr{M}(p)_{sr} = \begin{cases} 0 & \text{if } s = r,\ r \notin R(p), \\ u(p)_r - l(p)_r & \text{if } r \in R(p),\ s \in R(p), \\ \varepsilon & \text{otherwise.} \end{cases}$$

Calculus in the heap model are based on $(max,+)$-algebra. More precisely, if a piece p_1 is piled up in an empty heap, the upper contour of the heap is given by $x(p_1) = I \otimes \mathscr{M}(p_1)$ where I is a $1 \times |\mathscr{S}|$ matrix defined by $I_j = 0$, $\forall j \in \{1,\dots,|\mathscr{S}|\}$. In the same manner, the pile of two pieces p_1 and p_2 give the following upper contour of the heap : $x(p_1 p_2) = I \otimes \mathscr{M}(p_1) \otimes \mathscr{M}(p_2)$.

Example 2: Let us consider a 3 slots heap and 4 pieces a, b, c and d such that :

$$\mathscr{M}(a) = \begin{pmatrix} 2 & 1 & 2 \\ 2 & 1 & 2 \\ 2 & 1 & 2 \end{pmatrix}, \quad \mathscr{M}(b) = \begin{pmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 4 & \varepsilon \\ \varepsilon & \varepsilon & 0 \end{pmatrix}$$

$$\mathscr{M}(c) = \begin{pmatrix} 3 & 1 & 1 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{pmatrix}, \quad \mathscr{M}(d) = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 3 & 1 \\ 1 & 3 & 1 \end{pmatrix}$$

The pattern composed of the pile of the 4 pieces leads to

$$\mathscr{M}(acbd) = \mathscr{M}(a) \otimes \mathscr{M}(c) \otimes \mathscr{M}(b) \otimes \mathscr{M}(d) = \begin{pmatrix} 8 & 10 & 8 \\ 8 & 10 & 8 \\ 8 & 10 & 8 \end{pmatrix}$$

The upper contour of the heap is then

$$x_{\mathscr{H}}(acbd) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 8 & 10 & 8 \\ 8 & 10 & 8 \\ 8 & 10 & 8 \end{pmatrix} = \begin{pmatrix} 8 & 10 & 8 \end{pmatrix}.$$

The figure 4 illustrates the heap obtained.

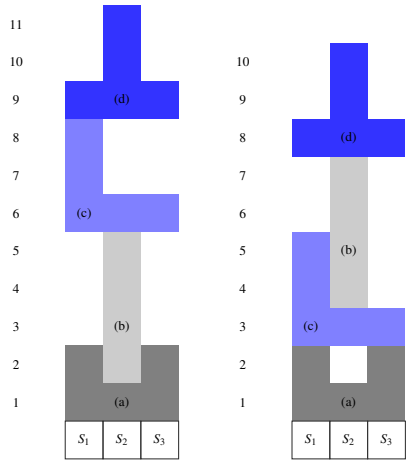

Fig. 4. Two patterns: *abcd* and *acbd*

IV. Application to the cyclic jobshop problem

A. Model construction

In this section, we show how to model the cyclic job shop problem as a heap of pieces. As shown in [8], the job shop problem admits a heap realization described by

$$\begin{array}{rcll} \mathscr{P} & = & \mathscr{T} & \\ \mathscr{S} & = & \mathscr{R} \cup \mathscr{J} & \\ R(t) & = & m_t \cup \{\mathscr{J}_i | t \in \mathscr{J}_i\} & \\ l(t,r) & = & 0 \text{ if } m_t = r, & l(t,r) = \varepsilon \text{ otherwise} \\ u(t,r) & = & p(t) \text{ if } m_t = r, & u(t,r) = \varepsilon \text{ otherwise.} \end{array}$$

In this framework, each elementary task t is represented by a matrix $\mathscr{M}(t)$. Considering the objective of the cyclic job shop problem, the problem is to find the sequence of pieces piled up in the heap that maximize the throughput. The following theorem indicates how to compute the cycle time

Theorem 1: The throughput of a job J_i for the cyclic sequence $p_1 \dots p_n$ in a heap model representing a job shop system is given by

$$\lambda_{J_i} = |p_1 \dots p_n|_{J_i} \otimes (\rho(\mathscr{M}(p_1) \otimes \dots \otimes \mathscr{M}(p_n)))^{-1}. \quad (6)$$

where $|p_1 \dots p_n|_{J_i}$ denotes the number of jobs J_i completed under the sequence $p_1 \dots p_n$ and $\rho(X)$ is the unique eigenvalue of the irreducible matrix X (see [12, chap.2]).

Proof: Considering the heap, the cyclic sequence $p_1 \dots p_n$ corresponds to a cyclic piled-up of these pieces. Let $v = p_1 \otimes \dots \otimes p_n$. The matrix $\mathscr{M}(v)$ can be seen as the matrix representing an unique piece $p_1 \dots p_n$. The evolution of the heap is now given by the pile up of v. As seen in section III, the upper contour of the heap is given by $x(v \dots v) = I \otimes \mathscr{M}(v) \otimes \dots \otimes \mathscr{M}(v)$. Since the job shop is connected (and we assume it is), the matrix $\mathscr{M}(v)$ is irreducible. After a certain number of iterations (that is called the transient

time), a cyclic behaviour appears for an irreducible matrix $X : X^{k+c} = \rho(X)^c \otimes X^k$ (see [17]). This property hold for $\mathscr{M}(v)$ and after a certain number of iterations, the growth rate of the heap become $\rho(\mathscr{M}(v))$ and the throughput is given by (6). ■

B. A branch and bound algorithm for the CJSP

We propose a branch and bound procedure to solve this problem. The branching rule chose the piece to pile up in the heap. We experimentaly observe that best results were obtained when the task with greatest processing time is selected first. Due to the particular structure of the matrix $\mathscr{M}(v)$

Then the evaluation function is the cycle time computation of the pattern in the heap. As remarked in [8], two tasks a and b belonging neither to the same job nor to the same machine can be piled up in an equivalent order since in this case $\mathscr{M}(a) \otimes \mathscr{M}(b) = \mathscr{M}(b) \otimes \mathscr{M}(a)$. This latter remark leads to an important reduction of the number of nodes in the search tree. Moreover, the following theorem derived two lower bounds from the structure of the problem (see [16]).

Theorem 2: Consider a CJSP, we have the following lower bound denoted α^- for the cycle time.

$$\alpha^- = \max\{\alpha_u, \alpha_s\}. \tag{7}$$

in which α_u is the minimum cycle time of the BCSP (the same problem without resource constraints) induced by the uniform constraints and α_s is defined below

$$\alpha_s = \max_{m \in \mathscr{R}}\{\sum_{i \in \mathscr{T} | m_i = m} p_i\}. \tag{8}$$

The bound α_u is significantly dependant on the WIP (see [13] and [16]). More precisely, we have

$$\alpha_u = \max_{J \in \mathscr{J}}\{\frac{\sum_{i \in J} p_i}{WIP}\}.$$

Proof: The lower bound α_s is the sum of all processing times of tasks belonging to a same machine and we obviously have

$$t_j(k+1) \geq \sum_{i \in T_j} p_i + t_j(k) \qquad \forall j \in \mathscr{T}$$

Since all tasks have the same cycle time, even if they do not belong to the same machine, we deduce that the cycle time of the CJSP cannot be smaller than the maximal sum of processing times per machine. In other words, α_s is deduced by the bottleneck machine. ■

Example 3: Consider again example 1 and assume WIP=2 we have $\alpha_u = max\{\frac{7}{2}, \frac{3}{2}, \frac{3}{2}, \frac{3}{2}\} = \frac{7}{2}$ and $\alpha_s = max\{5,5,6\} = 6$. We conclude that $\alpha^- = 6$.

C. Advantages and weaknesses of the heap of pieces model for the CJSP

Considering the heap of pieces model, the cyclic job shop problem is equivalent to find a bounded sequence of pieces with the maximum throughput. Compared to classical approaches, this model takes benefit from K-cyclic schedules. Indeed, graph based approach (such as the mixed integer linear program in II-B) have to consider $K \times n$ number of nodes. Regarding mixed integer linear programming, the number of variables grows substantially whereas the number of slots of the heap of pieces model remains the same since it is independant of K.

Nonetheless the drawback of this method is that we can only consider one work in progress and it can not represent a "nested" sechedule (as in Fig.2 for example) since it can only represent pattern that can be piled up (no intertwine).

Let us consider this very small example to illustrate this drawback.

Example 4:

Job	J_1		J_2	
Task	t_1	t_2	t_3	t_4
Processing time	5	4	2	3
Machine	M_1	M_2	M_1	M_2

Assume that we pile up $t_1t_2t_3t_4$, it leads to the heap in Fig. 5 and a cycle time $\alpha = 9$. It is obvious that the Gantt representation, depicted in Fig. 6 produced by the heap representation is not optimal.

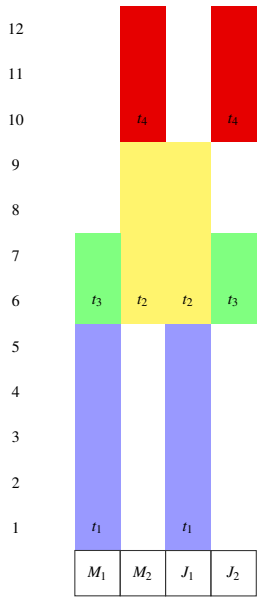

Fig. 5. Heap representation of the pattern $t_1t_2t_3t_4$

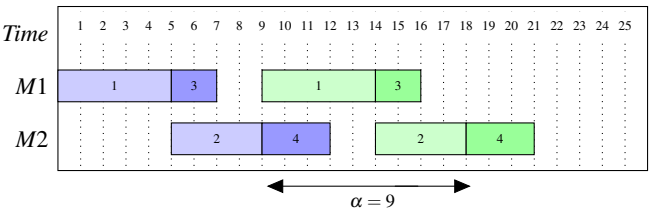

Fig. 6. Gantt diagram of the pattern $t_1t_2t_3t_4$

Besides, only one WIP is possible inside the pattern for $K-$cyclic schedules. If we consider the example 1, and a 2-cyclic schedule $w = t_1t_4t_4t_2t_8t_9t_6t_5t_3t_1t_8t_2t_7t_9t_5t_5t_6t_7$,

we obtain the heap depicted in figure 7 and we obtain $\rho(\mathscr{M}(v)) = 15$ and the cycle time is then $\alpha = \frac{15}{2}$. As we can remark, the second occurrence of J_1 doesn't start before the end of the first occurrence.

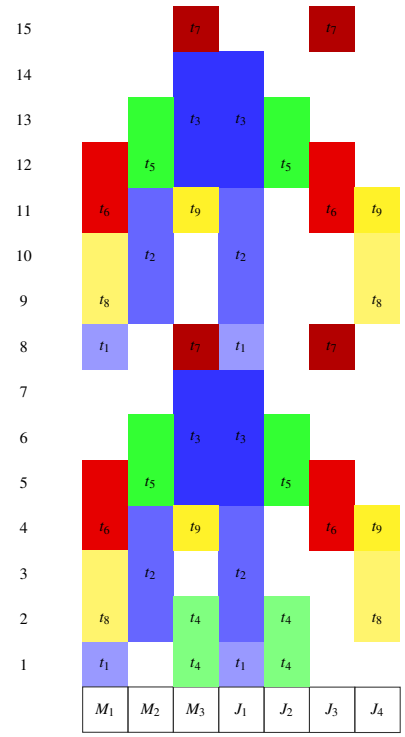

Fig. 7. Heap representation of the pattern $w = t_1t_4t_4t_2t_8t_9t_6t_5t_3t_1t_8t_2t_7t_9t_5t_5t_6t_7$

These two disadvantages limit the heap of pieces approach for scheduling. To deal with this situation, we propose in the next section an extension of this model to consider several works in process inside the pattern (but not nested pattern) through an increase of the number of slots.

D. Extension of the heap of pieces model

We extend this model to consider several work in process inside the pattern (but not nested pattern) through an extension of the number of slots.

Definition 2: A $K-$cyclic jobshop problem is specified by:

$$\begin{array}{rcll} \mathscr{P} & = & \mathscr{T} & \\ \mathscr{S} & = & \mathscr{R} \cup \mathscr{J}^K & \\ R(t) & = & m_t \cup \{\mathscr{J}_i | t \in \mathscr{J}_i\} & \\ l(t,r) & = & 0 \text{ if } m_t = r, & l(t,r) = \varepsilon \text{ otherwise} \\ u(t,r) & = & p(t) \text{ if } m_t = r, & u(t,r) = \varepsilon \text{ otherwise.} \end{array}$$

In this new model, several occurrences of a job can be processed simultenaously inside a pattern. More precisely, the slots of the heap are now composed by the machines and K Jobs. It enables K occurrences simltenaously of a job.

The size of the model increases as we have $|\mathscr{R}| + K \times |\mathscr{J}|$ slots. Let us consider the same schedule as in Fig. 7, the new representation leads to Fig. 8.

The evaluation of the cycle time in the new representation leads to $\alpha = \frac{13}{2}$ and we have the Gantt diagram in Fig. 9

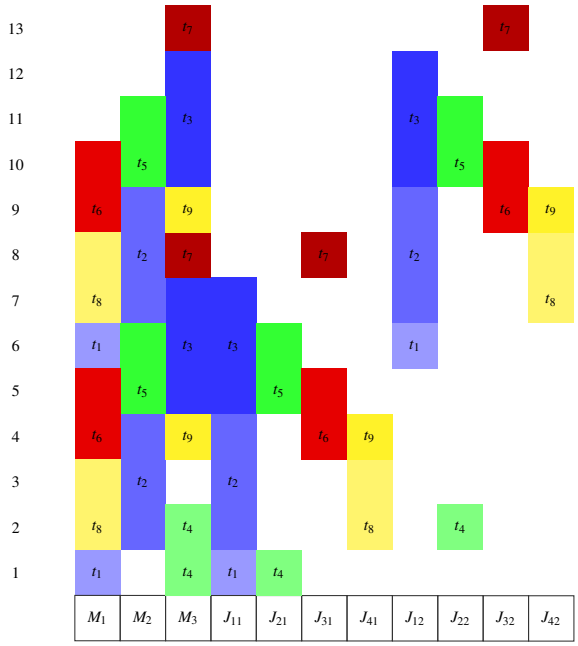

Fig. 8. Extended heap representation of the pattern $w = t_1t_4t_4t_2t_8t_9t_6t_5t_3t_1t_8t_2t_7t_9t_5t_5t_6t_7$

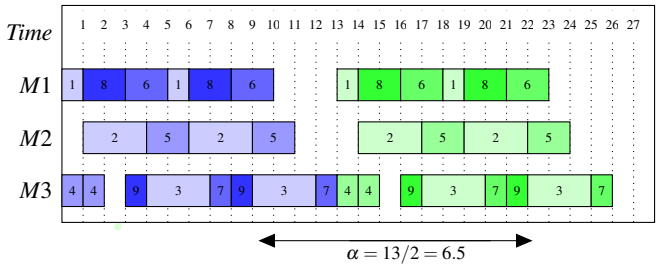

Fig. 9. Gantt diagram of the pattern of the pattern $w = t_1t_4t_4t_2t_8t_9t_6t_5t_3t_1t_8t_2t_7t_9t_5t_5t_6t_7$

V. Conclusion

We presented a method to model a solution of a cyclic jobshop problem. Moreover we exhibit the advantages and the weaknesses of the method. Indeed, the cyclicity and the work in process are two parameters that significantly defines the structure of the solution. Although the heap of pieces is not able to consider WIP greater than one, it reveals to be efficient for $K-$cyclic scheduling while graph based method are rapidly ineffective. An extension of heap of pieces model is proposed. It allows to consider more than one job in process inside the pattern. Consequently, this extension improve only $K-$cyclic schedules. Despite of the increase of the size of the model, it is worthwile to consider the extended model since it removes strong constraint on the the structure of the $K-$cyclic schedule.

Further works will compare the approach considered in this paper with the mixed integer linear program defined in §II-B on classical cjsp benchmark set.

REFERENCES

[1] V. Kats and E. Levner, "A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling," *Operations Research Letters*, vol. 21, pp. 171–179, 1997.

[2] M. Pinedo, *Planning and Scheduling in Manufacturing and Services*. Springer, 2005.

[3] H. P. Hillion and J. M. Proth, "Performance evaluation of job-shop systems using timed event graphs," *IEEE Transaction on Automatic Control*, vol. 34, no. 1, pp. 3–9, Jan. 1989.

[4] C. Hanen and A. Munier, "A study of the cyclic scheduling problem on parallel processors," *Discrete Applied Mathematics*, vol. 57, pp. 167–192, 1995.

[5] ——, *Scheduling Theory and Its Applications*. Wiley, 1995, ch. Cyclic Scheduling on Parallel Processors: An Overview.

[6] C. Hanen, "Study of a np-hard cyclic scheduling problem: The recurrent job-shop," *European Journal of Operational Research*, vol. 72, no. 1, pp. 82 – 101, 1994.

[7] P. Brucker and T. Kampmeyer, "A general model for cyclic machine scheduling problems," *Discrete Applied Mathematics*, vol. 156, no. 13, pp. 2561 – 2572, 2008.

[8] S. Gaubert and J. Mairesse, "Modeling and analysis of timed petri nets using heap of pieces," *IEEE Trans. on Automatic Control*, vol. 44, no. 4, Apr. 1999.

[9] M. Gondran and M. Minoux, *Graphes et algorithmes, 3e édition revue et augmentée*. Eyrolles, 1995.

[10] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Math*, vol. 23, 1978.

[11] A. Dasdan and R. K. Gupta, "Faster maximum and minimum mean cycle algorithms for system-performance analysis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, no. 10, 1998.

[12] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat, *Synchronization and Linearity*. Wiley, 1992.

[13] L. Houssin, "Cyclic jobshop problem and (max, plus) algebra," in *Proceedings of IFAC World Congress 2011*, vol. 18, no. 1, Milano, Italy, aug. 2011.

[14] R. Howard, *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology, 1960.

[15] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. M. Gettrick, and J. P. Quadrat, "Numerical computation of spectral elements in max-plus algebra," in *Proceedings of the IFAC Conference on System Structure and Control*, Nantes, 1998.

[16] M. Fink, T. B. Rahhou, and L. Houssin, "A new procedure for the cyclic job shop problem," in *Proceedings of INCOM 2012*, vol. 14, no. 1, Bucharest, Romania, may 2012.

[17] G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot, "Analyse du comportement périodique des systèmes de production par la théorie des dioïdes," INRIA, Le Chesnay, France, Rapport de recherche 191, 1983.