Chapter from the book *Petri Net, Theory and Applications*

Downloaded from:
http://www.intechopen.com/books/petri_net_theory_and_applications

Interested in publishing with IntechOpen?
Contact us at book.department@intechopen.com
Scheduling Analysis of FMS Using the Unfolding Time Petri Nets

Jong kun Lee¹ and Ouajdi Korbaa²
Changwon National University¹, Changwon
Ecole Centrale de Lille², Lille
Korea¹, France²

1. Introduction

FMS (flexible manufacturing system) is composed of a set of versatile machines, zig and fixture, and automatic transport system for moving parts between each job. In FMS, one of the important subjects is to formulate the general cyclic state-scheduling problem to minimize the WIP in order to satisfy economical constraints. Various methods have been proposed by researchers (Zuberek, 1987; Korbaa, 1997; Richard, 1998; Murata, 1989; Kondratyev, 1996; Hwang, 1997; Liu, 1999; Lee, 2004) to solve this problem. Hillion (Hillion, 1987) proposed a heuristic algorithm based on the computation of the degree of feasibility after giving a Petri net (PN) model of the system. Also, Valentin (Valentin, 1994) improved this algorithm by using Timed hybrid Petri nets and by introducing available intervals concept. The problem was that this algorithm could not guarantee the feasibility of the solution in one run. Korbaa (Korbaa, 1997) developed an algorithm to find near-optimal solution using the regrouping algorithm. Korbaa tried to get near-optimal solution and to minimize the WIP. For getting an optimal solution long computational time has been required. This means that all these methods have the problem of complexity and computational time. To simplify the calculation process in the scheduling problem and to make shorter computational time for getting many feasible solutions, we consider unfolding Petri nets to analyze the sequence process and to explain the reduced process. We call “unfolding” a PN unfolding, which has the reachability properties of the original net. Structural analysis on “unfolding” is much easier than on the initial model. The advantage of unfolding is that the state space explosion can be avoided since it is based on partial order semantics. To reduce the processing time, we consider an algorithm to select an environment of a shared resource, which has priority over the other ones, using the transitive matrix. In our case, the system has been analyzed to get the best possible solutions based on this environment of shared resource. The model has been divided into slices and create PN slice using this concept. The properties of a PN can be classified into categories: behavioral and structural properties. The behavioral properties are investigated in association with the marking of PN, e.g., reachability, boundness and liveness (Liu, 1999). Both transition and place invariants belong to structural properties in PN. If the behavioral
properties used by the transitive matrix are exhibited, it could be easier to analyze the system after slicing off some subnets.

This paper is organized as follows: some definitions of Time Petri Nets (TPN) and unfolding are given in Section 2, and time Petri net slice is presented in Section 3. The scheduling objectives and outline the problems arising during the transformation of the initial model into an unfolding TPN are described in Section 4. In Section 5, we introduce an illustration model for FMS, and specifically emphasize the different ways of obtaining a closed loop model. In Section 6, we are showing the benchmarking resultants after analyzing the performance evaluation factors. A conclusion and some perspectives will end this paper.

2. Basic notions of time Petri nets and slices

In this section certain terms that are often used in this paper are defined (Best, 1990; Carlier, 1988; Camus, 1997; Esparza, 1996; Julia, 1995; Lee, 1995; Lee, 2004; Lee, 2006; Liu, 1999; Murata, 1987; Taubin, 1997).

Let \( N = \langle P, T, I, O, M_0, \tau \rangle \) be a Time Petri Nets, where \( P \) is the set of places (\( m \) elements), \( T \) is the set of transitions (\( n \) elements), and \( P \cap T = \emptyset \). \( I: T \rightarrow P \) is the input function, \( O: T \rightarrow P \) is the output function. \( M_0 \in M = \{ M \mid P \rightarrow N \} \), \( M_0 \) is an initial marking, \( N \) is the set of positive integers. \( \tau: T \rightarrow N \) is a time function which associates to each transition of \( T \) a deterministic rational.

A transition \( t \in T \) is enabled at a marking \( M' = M - t + \tau \), where \( t \) represents the incoming weights of the fired transition and \( \tau \) the outgoing weights of the fired transition. We denote by \( M \) (\( t \rightarrow M' \)). The set of reachable marking of \( N \) is the smallest set \( \{ M_0 \} \) and such that if \( M \in \{ M_0 \} \) and \( M \rightarrow M' \) (for some \( t \in T \)) then \( M' \in \{ M_0 \} \) and bounded if there is \( k \in N \) such that \( M(P) \subseteq [0,\ldots, k] \), for every reachable marking \( M \). The set of successors of a node \( x \in P \cup T \) is \( x = \{ y \in P \cup T \mid (x, y) \in F: I \cup O \} \), and the set of its predecessors is \( x = \{ y \in P \cup T \mid (y, x) \in F: I \cup O \} \).

A Time Petri nets \( N = \langle P, T, F, M_0, \tau \rangle \), where \( P \subseteq P, T \subseteq T, F \subseteq F, M_0 \in \{ M_0 \} \) and \( \tau \in \tau \) then we can say that \( N \) is sub-net of \( N \).

The number of occurrences of an input place \( P_i \) in a transition \( t_j \) is given by \( \#(P_i, I(t_j)) \), and the number of occurrences of an output place \( P_i \) in a transition \( t_j \) is given by \( \#(P_i, O(t_j)) \).

The matrix of the PN structure, \( S = \langle P, T, C_- , C_+ \rangle \), where \( P, T \) are the finite set of places and transitions, respectively. \( C_- \) and \( C_+ \) are matrices of \( m \) rows (one for each place) and \( n \) columns (one for each transition) defined by:

\[ C_- = [I,j] = \#(P_i, I(t_j)), \text{ matrix of input function}, \]

\[ C_+ = [O,j] = \#(P_i, O(t_j)), \text{ matrix of output function}. \]

And the incidence matrix \( C \) is given by \( C = C_+ - C_- \).

3. Invariant and transitive matrix

3.1 Invariant

For completeness, we recall the terminologies which were used in (Lee, 2001; Lee, 2004; Lee, 2006; Liu, 1999).
(Def. 3.1): Invariant
A row vector \( X = (x_1, x_2, \ldots, x_n) \geq 0 \) is called a \( P \)-invariant if and only if \( X \cdot C = 0 \), where \( x \neq 0 \) and \( \cdot \) denotes the vector product.
A column-vector \( Y = (y_1, y_2, \ldots, y_n)^T \geq 0 \) is called a \( T \)-invariant, if and only if \( C \cdot Y = 0 \), where \( Y \) is an integer solution of the homogeneous matrix equation and \( Y \neq 0 \).

(Def. 3.2): Place transitive and Transition matrix
The place transitive and transition matrix are defined, respectively as follows:

\[
C_P = C(C^+)^T \\
C_T = (C^+)^T C.
\]

(Example):

\[
C_P = \begin{bmatrix}
0 & 1 \\
1 & 0 \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

Fig. 1. A Petri net
This matrix allows representing the relation between places in terms of output and input. For previous example, we can find that \( p_2 \) and \( p_3 \) receive a token after \( p_1 \) and that \( p_1 \) receives one token from \( p_2 \) and another from \( p_3 \) (Fig. 1).

(Def. 3.3): Labeled place transitive matrix
Let \( L_{CP} \) be the labeled place transitive matrix:

\[
L_{CP} = C^- \text{diag}(t_1, t_2, \ldots, t_n)(C^+)^T
\]

The elements of \( L_{CP} \) describe the direct transferring relation that is from one place to another one through one or more transitions.

(Def. 3.4): Let \( L_{CP} \) be the \( m \times m \) place transitive matrix. If a transition \( t_k \) appears \( s \) times in the same column of \( L_{CP} \), then we replace \( t_k \) in \( L_{CP} \) by \( t_k / s \) in \( L^*_{CP} \).

Through introducing the \( m \times m \) place transitive matrix, we can evaluate the transition enabling firing, and calculate the Quantity and the Sequence of Transition enabling Firing.
(Def.3.5): Let a reachable marking $M_R(K + 1)$ from $M(k)$ be an $m$-vector of nonnegative integer. The transformation is defined by:

$$M_R(k + 1)^T = M(k)^T L_{CP}^*$$

**Fig. 2. Illustrative example**

(Example):
This example net explained like that $M(k) = [P_1(k), P_2(k), P_3(k)]^T$, then we can obtain (Fig. 2):

$$L_{BP} = \begin{bmatrix}
0 & 0 & t_1 \\
0 & 0 & t_2 \\
0 & 0 & 0
\end{bmatrix}$$

and $M_R = [0, 0, t_1(P_1(k)) + t_2(P_2(k))]^T$.

### 3.2 Algorithm and properties

A basic Unit of Concurrency (BUC) is a subnet corresponding to a resource. It contains the incoming transitions of the resource and also the outgoing transitions. We chose the BUC based on the place transitive matrix; in this paragraph we propose an algorithm to determine the BUC.

**Algorithm: BUC construction algorithm**

Input: $N = <P, T, I, O, M_0, \tau>$

Output: BUC of $N$, $N_S = <P_S, T_S, I_S, O_S, M_0, \tau_S>$

Define $L_{CP}$ and consider one shared resource (machine)
1. Find the all-relational places, of the shared resource, in each column and row \( L_{CP} \) and make an element of own BUC with this initial marking place.

2. Find the relational place of selected place in (1).

3. Link all selected places and transitions with existing arcs on the initial Petri net.

The method of partitioning the model divides the system into BUC. The obtained BUC is defined as follows:

(Def.3.6): BUC

In Time Petri net \( N= <P, T, I, O, M_0, \tau> \), when the places set is divided by the previous algorithm, the BUCs are defined by \( \{BUC_i \mid i=1,\ldots,n\} \), and each \( BUC_i = (P_i, T_i, F_i, M_i, W_i) \) satisfies the following conditions.

\[
P_i = P_{BUC_i}
\]

\[
T_i = \{ t \in T \mid s \in P_i, (s, t) \in F \text{ or } (t, s) \in F \},
\]

\[
F_i = \{ (p, t) \in F, (t, p) \in F \mid p \in P_i, t \in T_i \},
\]

\[
\forall \tau_i \in \tau, \tau_i(t) = \tau(t) \text{ and } \forall p \in M_i, M_i(p) = M(p).
\]

We can say that the flow of control in the Petri nets is based on the tokens flow. If the token is divided into some tokens after firing a transition, the flow of control divides to several flows. Accordingly, we define that the BUC is a set of the executed flow of control based on the functional properties of the net. In the Time Petri net model, the behavioral condition of transition \( t \) are all predecessors’ places (*t), which are connected with transition \( t \). Petri net Slices are subnets based on BUC at the transition level, and a functional condition is defined according to the following conditions.

- When transition \( t \) is not shared: satisfy the precondition of transition \( t \) only.
- When several slices share transition \( t \): satisfy the preconditions in all BUC, which have transition \( t \).

(Theorem. 3.1) Let \( N \) be a Time Petri net and \( N_s \) be a set of Time Petri net slices that is produced by the slice algorithm. If \( N_s \) satisfies the functional conditions, \( N \) and \( N_s \) are behavioral equivalent \( (N \sim N_s) \).

(Proof) (We prove this theorem based on the \( p \in \bullet t_i \Leftrightarrow p \in \bullet t \).

\( (\Rightarrow) \) Since the \( p \) is an element of \( P_i \), \( \exists t_i \in T_i \) such that \( p \in \bullet t_i \). And by the BUC definition, \( P_i = P_{BUC_i} \). \( P_{BUC_i} \) is a set of place which is obtained by the BUC algorithm, such that \( P_{BUC_i} \subseteq P \). So, we say that if \( p \in \bullet t_i \) then \( p \in \bullet t \) is true.

\( (\Leftarrow) \) Consider if \( p \in P \) and \( p \notin P_{BUC_i} \) then \( P \nsubseteq P_{BUC_i} \). If \( p \in P \), \( t \in T \) such that \( p \in \bullet t \). And \( p \notin \bigcup_{t \in T_i} \bullet t_i \), in this case, we can say that \( \bigcup_{t \in T_i} \bullet t = \bigcup_{t \in T_i} P_i \), but by the definition of BUC, \( P_i \) is a set of places which is obtained by the BUC algorithm, such that \( P_i \subseteq P \). So if \( p \in P \) and \( p \notin P_{BUC_i} \) then \( P \nsubseteq P_{BUC_i} \) is not true. \( \square \)

4. Unfolding time Petri nets (UTPN)

Unfolding technique, originally proposed by McMillan(McMillian,1995), is a method used to avoid the state explosion problem in the verification of concurrent systems modeled with finite-state Petri nets. The technique is based on the concept of net unfolding; well-
unknown partial order semantics of Petri nets (Hwang, 1997; Kondratyev, 1996; Lee, 2001; McMillian, 1995).

4.1 Occurrence Nets
An occurrence net is a net, which represents clearly causal relations between places, especially transitions, of the original net. In this section, we briefly recall the main definitions (Fig. 3 and 4).

Fig. 3. Cyclic Petri net

Fig. 4. Occurrence net of Fig. 3.
(Def.4.1) (OCN (Occurrence net)). An occurrence net is an acyclic Time Petri net \( N = \langle P, T, F, M, \tau \rangle \) in which every place \( p \in P \) has at most one input transition \( (\forall p \in P, F(\langle T \times P \rangle)) \) is the flow relations.

An OCN algorithm is summarized as follows (Hwang, 1997; McMillian, 1995):

**Algorithm: (Occurrence net construction)**

**Input:** \( N = \langle P, T, F, M, \tau \rangle \)

**Output:** the OCN of \( N, N' = \langle P', T', F', M', \tau' \rangle \).

1. Make a copy of the place in to the OCN and labeled as \( p' \).
   Repeats (2)-(6) until the transitions set becomes empty.

2. Choose a transition \( t \in T \).
3. For each place in \( \cdot t \), find a copy in the OCN, and if not found then go back to (2).
   Do not choose same subnet in OCN twice for a given \( t \).
4. If any pair of chosen places is not concurrent, go to (1).
5. Make a copy of \( t \) in OCN and labeled as \( t' \). Draw an arc from every places which was chosen to \( t' \).
6. For each place in \( t \) in OCN and labeled as \( t' \). Draw an arc from every places which was chosen to \( t' \).

4.2 Unfolding

Unfolding is used to verify the occurrence net after cut or truncate, based on local configuration and basic marking (Kondratyev, 1996; McMillian, 1995).

(Def.4.3) (Configuration). A set of transitions \( C \subseteq T' \) is a configuration in an OCN if:

1. for each \( t \in C \) the configuration \( C \) contains \( t \) together with all its predecessors;
2. \( C \) contains no transitions in mutual conflict.

(Def.4.4) Let \( C' \) be a configuration of an occurrence net. A final marking of \( C' \), denoted by \( FM(C') \), is a marking reachable from the initial marking after all transitions of \( C' \) and only those transitions are fired. A final marking of a local configuration of \( t' \) of the \( C' \) is called the local configuration of \( t' \) and is denoted as \( \tau(t') \).

(Def.4.5) (Cut off). A transition \( t' \) of an occurrence net is a cut off transition, if there exists another transition \( t'' \) such that \( BM(t'/t'') = BM(t'') \) and \( |\tau(t')| > |\tau(t'')| \), where \( |\tau(t')| \) is the cardinality of \( \tau(t') \). An unfolding is the greatest backward closed subnet of an occurrence net containing no nodes after cut-off transition.

(Def.4.6) An unfolding time Petri net \( UTPN = \langle P', T', F', M', \tau' \rangle \) is obtained from the occurrence net by removing all the places and transitions, which succeed the cut-off (Fig. 5).
Let UTPN be a unfolding Time Petri net, Mo be the initial marking and FM be a final marking, find a sequence x such that: BM (x>FM, and the reachability delay is minimal (Richanrd,1998):

$$\min \ z = \sum_{i \in T} x_i$$

Subject to BM (x>FM where X=(xₜ) is the characteristic vector of the sequence x.

5. Modeling of UTPN

An important sub-class (sliced net) of Time Petri nets is a net in which has an independent control flow and for which all weights associated to arcs are equal to one. In this works, we select an independent control flow based on the machines operations. This Time Petri net can perfectly model the command we want to implement. At the end of the optimization approach we obtained a model where all the operating sequence are linear and the next step is to compute the best schedule.

5.1 Computing the optimal schedule

The schedule of a sliced net can be easily computed by playing the token and firing transition as soon as possible. We can compute firing dates of transitions by computing potentials on the sliced net after unfolding. In the unfolding net UTPN, it has one function for compute the makespan time, as $f(UTPN)$. The function $f(UTPN)$ is the necessary time to go from the initial marking Mo to the object marking. $f(UTPN)$ is composed of $h(t_i)$ and $g(t_i)$, where $h(t_i)$ is an operating time of transition $t_i$ and $g(t_i)$ is an operating time of the next transition $t_j$ (we call it minimum waiting time to fire transition $t_j$). Then, the schedule duration can be computed by the following recurrent formula:

$$f(UTPN) = h(t_i) + g(t_i)$$

and
The best schedule of the UTPN is obtained by minimizing the function \( f(UTPN) \). We can say that if we find a minimized unfolding net, the schedule of marking is an optimized schedule. So let UTPN be the makespan time of the optimized unfolding schedule. Then the time can be computed using the unfolding net as follows:

\[
\text{Optimize } UTPN = \min \{ f(UTPN) \}
\]

To apply our method continuously, we consider some notations about degree of operation time in the machine and minimized WIP. The deciding order is one of the important things in the scheduling problem. We consider a throughput of the operation time in the machine, based on the number of tokens (number of resource share), and the operating time of machine.

Let \( d(m) \) be the degree of operation time in the machine, then

\[
d(m) = \frac{\phi(m)}{\gamma(m)}
\]

where, \( \phi(m) \) is total operation time of the machine \( i \), and \( \gamma(m) \) is number of token (resource share transition) in machine \( i \). So, this degree of \( d(m) \) is one parameter to choose the select an order to apply in Unfolding net.

In the linear cyclic scheduling problem, the minimization of the number of pallets is one of the important factors.

(Def. 5.1) Let CT be an optimal cyclic time based on the machines work, then WIP lower bound is:

\[
WIP = \left\lceil \frac{\sum_{i=1}^{M} \sum_{O(i)\text{subcycledby}} \text{Operatingtime}}{CT} \right\rceil
\]

### 5.2 Dispatching rules

We consider a system with two machines such as M1, M2 and two jobs such as OP1 and OP2 in (Camus, 1996). Let us assume that the production ratio corresponds to the production of 50% of OP1, 50% of OP2 (Fig. 6).

Now, we show the transitive matrix of the example as shown in Table 1. For simplicity reason, we ignore two places \( w_1, w_2 \) and one transition \( tw \). In this table, initial tokens exist in M1 and M2. The cycle time of OP1 and OP2 are 11 and 9, and the working time of machine M1 and M2 are both 10, respectively. So the cycle time CT is 10. The minimized WIP is:

\[
WIP = \frac{11 + 9}{10} = 2
\]

Also, about the degree of the feasibility time of the machine, M1 and M2 have same priority,
These machines have same operating time (i.e. 10) for three operations showing that it is not important to give the priority for first approach. So we select M1 to start with.

Fig. 6. A two shared resources example

Table 1. Transitive matrix of the illustration example

Now, we select row and column of p1, p3, p5, and M1 in (Table 1), and make one slice net based on the selected places and transitions.

1. Slice BUC of M1 and its unfolding nets

Machine M1 involved two tasks (OP1 and OP2) in three processes (t1, t3, t5)(Fig. 7 and 8).
Chapter Scheduling Analysis of FMS Using the Unfolding Time Petri Nets

Fig. 7. The five BUCs of M1


Fig. 8. Example of the unfolding of M1
In this net, we can find the 6 processes of M1 are as follows (Fig. 9):

- Suf1 = t1 t5 t3 (15),
- Suf2 = t5 t1 t3 (15),
- Suf3 = t1 t3 t5 (12),
- Suf4 = t3 t1 t5 (12),
- Suf5 = t3 t5 t1 (15),
- Suf6 = t5 t3 t1 (15),

where () is an operation time of Sufi.

In M1, we can choose two schedules as transitions sequences: t3-t1-t5 and t1-t3-t5.

2. Modeling of M2 and its unfolding nets

Machine M2 involved two tasks (OP1 and OP2) in three processes (t2, t4 and t6) (Fig. 10,11).
Fig. 11. Example of unfolding of M2

We can show the six processes like as follows (Fig. 12):

\[ \text{Suf}_1 = t_2t_4t_6 \quad (13), \quad \text{Suf}_2 = t_2t_6t_4 \quad (14), \quad \text{Suf}_3 = t_4t_6t_2 \quad (11), \]
\[ \text{Suf}_4 = t_4t_2t_6 \quad (13), \quad \text{Suf}_5 = t_6t_4t_2 \quad (11), \quad \text{Suf}_6 = t_6t_2t_4 \quad (13). \]

Fig. 12. Results of permutations of BUC in M2
In M2, we can find two solutions like as Suf3 and Suf5. Now, we apply the selected solutions of BUC of M2: {Suf3 and Suf5} to obtain the solution BUC on M1: {Suf3 and Suf4}, then we obtained two solutions. The optimal schedules of two cycles are in Fig. 13 and 14.

Bs1: A3A1B2 (M1) B3B1A2 (M2)
Bs2: A1A3B2 (M1) B3B1A2 (M2)
Chapter Scheduling Analysis of FMS Using the Unfolding Time Petri Nets

(a) Linear schedule
Fig. 15. Linear schedule of Bs1

(b) The flow of marking of (a)

(a) Linear schedule
Fig. 16. Linear schedule of Bs2

(b) The flow of marking of (a)
Finally, we get three pallets rather than two, which is lower bound WIP. Indeed in this model, it is impossible to optimize CT with two pallets, as proved in (Camus, 1997). So, we can say that this solution is best possible one (Fig. 15, 16).

6. Benchmark

6.1 Notations

In this section, one example taken from the literature is analyzed in order to apply three cyclic scheduling analysis methods such as Hillion (Hillion, 1987), Korbaa (Korbaa, 1997), and the previously presented approach. The definitions and the assumptions for this work have been summarized (Korbaa, 1997).

The formulations for our works, we can summarize as follows:

\[ \mu(\gamma) = \sum_{t \in \gamma} D(t), \]

the sum of all transition timings of \( \gamma \)

\[ M(\gamma) = \text{Mo}(\gamma), \]  
the (constant) number of tokens in \( \gamma \)

\[ C(\gamma) = \frac{\mu(\gamma)}{M(\gamma)}, \]

the cycle time of \( \gamma \),

Where \( \gamma \) is a circuit.

\[ C^* = \max(C(\gamma)) \]

for all circuits of the net,

\[ \text{CT} \]

the minimal cycle time associated to the maximal throughput of the system:

\[ \text{CT} = \max(C(\gamma)) \]

for all resource circuits = \( C^* \)

Let \( \text{CT} \) be the optimal cycle time based on the machines work, then WIP is (Korbaa, 1997):

\[ \text{WIP} = \sum_{\text{pallet type } i} \left[ \frac{\sum \text{Operating time of OS to be carried by } i}{\text{CT}} \right] \]

We introduce an illustrative example in Camus (Camus, 1997), two part types (P1 and P2) have to be produced on three machines U1, M1 and M2. P1 contains three operations: u1(2 t.u.) then M1(3 t.u.) and M2(3 t.u.) P2 contains two operations: M1(1 t.u.) and U1(2 t.u.). The production horizon is fixed and equalized to E={3P1, 2P2}. Hence five parts with the production ratio 3/5 and 2/5 should be produced in each cycle. We suppose that there are two kinds of pallets: each pallet will be dedicated to the part type P1 and the part type P2. Each transport resource can carry only one part type. The operating sequences of each part type are indicated as OS1 and OS2. In this case, the cycle time of OP11, OS12 and OS13 are all 7 and Op21 and OS22 all 3, also the machines working time of U1 is 10, M1 is 11 and M2 is 6. So the cycle time CT is 10. The minimization WIP is:

\[ \text{WIP} = \left[ \frac{\sum \text{Operating Times of OS}_{P1}}{\text{CT}} \right] + \left[ \frac{\sum \text{Operating Times of OS}_{P2}}{\text{CT}} \right] \]

\[ = \left[ \frac{7 + 7 + 7}{11} \right] + \left[ \frac{3 + 3}{11} \right] = 3 \]
Fig. 17. Illustrative example

6.2 Benchmark
By the example, we can obtain some results like as the following figures (Fig. 18-20).

1. Optimization
The Hillion’s schedule (Hillion, 1987) has 6 pallets, the Korbaa’s schedule (Korbaa, 1997) 3 ones, and the proposed schedule 4 ones. This solution showed that the good optimization of Korbaa’s schedule could be obtained and the result of the proposed schedule could be better than that of the Hillion’s.

Also, the solutions of the proposed approach are quite similar to (a) and (c) in Fig. 20 without the different position.

2. Effect
It’s very difficult problem to solve a complexity value in the scheduling algorithm for evaluation. In this works, an effect values was to be considered as the total sum of the numbers of permutation and of calculation in the scheduling algorithm to obtain a good solution. An effected value of the proposed method is 744, i.e. including all permutation available in each BUC, and selecting optimal solution for approach to next BUC. An effect value to obtain a good solution is 95 in the Korbaa’s method; 9 times for partitions, 34 times
for regrouping, and 52 times for calculation cycle time. In the Hillion’s method, an effected value is 260; 20 times for machine’s operation schedule and 240 times for the job’s operation schedule.

Fig. 18. Hillion’s schedule

Fig. 19. Korbak’s schedule
Based on the three algorithms, we can get time results for obtaining the good solution. Since this example model is simple, they need very small calculation times; 1 sec for the Korbaa’s approach and 1.30 sec for both of the Hillion’s and the proposed approaches. The Korbaa’s approach has minimum 1 minute and maximum 23 hours in the 9 machines and 7 operations case in Camus (Camus, 1997), while the proposed approach 3 minutes. Meanwhile the Hillion’s and the Korbaa’s approaches belong to the number of the operation and the machines, the proposed method to the number of resource shares machines. This means that the Hillion’s and the Korbaa’s approaches analyzing times are longer than the proposed one in the large model. As the characteristic resultants of these approaches are shown in Fig. 21, the Korbaa approach is found out to be good and the Hillion approach is
to be effectiveness in the time. And on the effort point, the proposed approach is proved to be good.

7. Conclusion and future study

In this paper, we focused on the analysis of a cyclic schedule for the determination of the optimal cycle time and minimization of WIP (Work In Process). Especially, this paper product ratio-driven FMS cyclic scheduling problem with each other products and ratios has been dealt. We proposed a model that has two jobs and two machines. And TPN slice and unfolding are applied to analyze this FMS model. We can divide original system into subsystem using TPN slice and change iterated cycle module into acyclic module without any other behavior properties.

Specially, we simulated our approach with IBM PC windows 2000 using Visual C++, then our approach is faster than Korbaa's approach in the many resource shared. This means that the new approach is more useful to the model that has many resource share machines in any case. If the model has small resource share machines and short operation depths, then it's useful to approach Korbaa's.

We are sure that proposed method is very useful to analyze all Petri net models. This proposed method is available to apply to a complex computer simulation, a parallel computer design and analysis, and a distributed control system, etc.

8. References


Although many other models of concurrent and distributed systems have been developed since the introduction in 1964 Petri nets are still an essential model for concurrent systems with respect to both the theory and the applications. The main attraction of Petri nets is the way in which the basic aspects of concurrent systems are captured both conceptually and mathematically. The intuitively appealing graphical notation makes Petri nets the model of choice in many applications. The natural way in which Petri nets allow one to formally capture many of the basic notions and issues of concurrent systems has contributed greatly to the development of a rich theory of concurrent systems based on Petri nets. This book brings together reputable researchers from all over the world in order to provide a comprehensive coverage of advanced and modern topics not yet reflected by other books. The book consists of 23 chapters written by 53 authors from 12 different countries.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following: