Deadlock Detection and Avoidance Algorithm in Petri Nets Using the Resource Sharing Matrix

Jong Kun Lee[1], Sang Hwan Kim[2]

[1]LIS/Computer Engineering Dept, Changwon National University,

Salim-dong 9, Changwon, Kyungnam, Korea

Email: jklee@changwon.ac.kr

[2] CIO/executive director, Daehan Calsonic Co.,

Ho-tang li 340, Ip-jang myun, Chunan, Chungbuk, Korea

E-mail: sanghwan@dhc.co.kr

Abstract: This paper considers deadlock detection problem for FMS (Flexible Management System) based on the relationship of the resource share places in Petri Nets model. Since a deadlock is a condition in which the excessive demand for the resources being used by others causes activities to stop, it is very important to detect and prevent a deadlock. In this paper, after analyzing the relation of resource-share places in Petri Nets, we study the deadlock condition in FMS. This paper intends to review and compare these deadlock detection and avoidance methods based on the complexity, the effect value and the algorithm understanding.

Keywords: avoidance, benchmark, DAPN, deadlock, Petri-nets, resource sharing, siphon, transitive matrix

Lee Jong-kun has been a professor and Ph.D. supervisor at Changwon National University (CNU), Korea since 1983. He received his B. Sc. degree in Computer Science from Soongsil University, and the M. Engineering degree in Computer Engineering from Soongsil University, MBA from the Korea University, and Ph.D. in Computer engineering from the Ecole Centrale Paris. He is also director of NEXUS center of CNU, and chairman of Kyungnam Smart home technique Forum in Kyungnam. Currently, his research interests include concurrent model analysis and performance evaluation, Scheduling analysis in FMS, Mobile security and Petri nets theory.

Kim Sang-hwan is a Ph.D. in Computer Engineering from Chungbuk National University, Korea. He is also general manager of Daihan Calsonic Co., manufacturing air-condition system of Nissan Motor Company in Korea. His research interests include Petri net, Semantic Grid and workflow theory.

1. Introduction

While deadlock problem is one of the important properties for analyzing the Petri nets, at the same time, it also could be one of the critical points in the scheduling problem of FMS (flexible Management System). Deadlock is a stopping status of the flow of a marking due to a resource waiting. Deadlock usually appears in contain subsystem which is run in parallel and resources share places.

The approaches to address deadlock are classified as prevention, avoidance, and detection/resolution. Deadlock *prevention* is a static server allocation approach, and averts deadlock by ensuring that the necessary conditions for deadlock cannot occur. Examples of preventive measures in a manufacturing system include uni-directional batching of jobs and providing a large number of in-process buffers. Deadlock *avoidance* is a dynamic server allocation approach that prevents the sufficient conditions for deadlock from occurring. Also, deadlock detection/resolution is a recovery procedures which detect deadlock occurs and restore the system operation [20]. Most of technical literatures have been used Petri nets to derive deadlock prevention and avoidance algorithms [1-3, 6, 9-11,13-19]. Deadlock avoidance methods are in many cases related to a set of sequential process sharing common resources of [15]. Xing et al. proposed a necessary and sufficient liveness condition based on the deadlock structure [16]. Moreover, J.park and S.A. Reveliotis [19] present deadlock avoidance policies for system with multiple resource acquisition and flexible routing, called conjunctive/disjunctive resource

allocation system. Control places were also used for siphon control in [7,8,15]. However, while many researches on an application of the DES (Discrete Event System) to need a real time process have been studied, the proposed studies failed to ease an understanding and application of a system included many processes. Our method for deadlock prevention is based on the structural property. It considers Petri nets component such as transition and place invariant. Since this method may detect the deadlock status and avoid schedule in the transitive matrix directly, it's easy to find the deadlock property. In this paper, we propose a method to analyze the deadlock problem and to avoid problem in Petri nets using the transitive matrix. Also, this paper is an extended variant of a work in [18].

This paper is organized as follows; some definitions of Petri Nets and transitive matrix are given in section 2. In section 3, we show a deadlock detection algorithm, and show an illustration model with one example. In section 4, their performance is analyzed and evaluated. Finally, in section 5, concludes the paper and suggests directions for future study.

2. Notations

In this section certain terms which are often used in the latter part of this paper are [4,7,8,17].

A Petri net is a 5-tuple PN=<P,T,I,O,M,> where $P = \{p_1,p_2,..,p_m\}$ is a finite set of places, $T=\{t_1,t_2,\ldots,t_n\}$ is a finite set of transitions, and $P \cup T = \varnothing$, $P \cap T = \phi$, I:P x T -> ℵ is an input function, O:T x P -> ℵ is an output function where ℵ is the set of positive integers. M:P-> ℵ, is a marking representing the number of tokens in places with Mo denoting the initial marking.

A transition $t \in T$ is enabled under M, in symbols M[t>, iff $\forall p \in {}^\bullet t: M(p) > 0$ holds. If M[t> holds the transition t may fire, resulting in a new marking M', denoted by M[t> M', with M'(p) = M(p) -1, if $p \in {}^\bullet t \backslash t^\bullet$; M'(p)=M(p) +1 if $p \in t^\bullet \backslash {}^\bullet t$; and otherwise M(p)=M'(p), for $\forall p \in P$.

The set of reachable marking from a marking Mo is denoted as R(PN,Mo).

Let (PN,Mo) be a Petri net with PN=<P,T,I,O,M>. A transition $t \in T$ is live under Mo iff $\forall M \in R(PN,Mo)$, $\exists M' \in R(PN,M), M'[t>$ holds. PN is dead under Mo iff $\nexists\ t \in T$, Mo[t> holds. (PN,Mo) is deadlock-free iff $\forall M \in R(PN,Mo), \exists t \in T, M[t>$holds. (PN,Mo) is quasi-live iff $\forall t \in T$, $\exists M \in R(PN,Mo)$, M[t> holds. (PN,Mo) is live iff $\forall t \in T$, t is live under Mo. (PN,Mo) is bounded iff $\exists k \in N, \forall M \in R(PN,Mo), \forall p \in P, M(p) \leq k$ holds.

The matrix of a PN structure, C is $C=<P,T,B^-,B^+>$, where $P = \{p_1,p_2,..,p_m\}$ is a finite set of places, $T=\{t_1,t_2,\ldots,t_n\}$ is a finite set of transitions, and $P \cup T = \varnothing$, $P \cap T = \phi$, . B^- and B^+ are matrices of m rows by n columns defined by

$B^- = [I,j] = \#(P_i,I(t_j))$, matrix of input function,

$B^+ = [I,j] = \#(P_i,O(t_j))$, matrix of output function.

Also, $B = B^+ - B^-$ is called an incidence matrix.

A column-vector $x^T=(x_1,x_2,\ldots,x_n) \geq 0$ of the homogeneous equation $A\,x^T=\Delta M =0$ is called a T-invariant, where x^T is x's transpose. An integer solution $y = (y1,y2,\ldots,ym)^T$ of the transposed homogeneous equation Ay = 0 is called a S-invariant.

The place and the transition transitive matrix are as follows, respectively:

$B = B^-(B^+)^T$: place transitive matrix

Let L_{BP} be the labeled place transitive matrix:

$L_{BP} = B^{-}\ diag(t_1, t_2, ..., t_n)(B^{+})^{T}$,

where $t_i (i = 1,2,...,n)$ is :

$$|t_i| = \begin{cases} 1 \text{ fire } t_i \\ 0 \text{ not fire } t_i \end{cases}$$

The elements of L_{BP} describe the directly transferring relation that is from one place to another place through one or more transitions.

Through introducing the m×m place transitive matrix, we can evaluate transition enabling firing, and calculate Quantity and Sequence of Transition enabling Firing.

Let a reachable marking $M_R(K+1)$ from $M(k)$ be an m-vector of nonnegative integer. The transformation defined by

$M_R(k+1)^T = M(k)^T M_{PR}$

The elements of M_{PR} describe the direct transferring relation that is from one place to another place through one or more transitions.

Let M_{PR} be the labeled place transitive matrix. If a transition t_k appears s times in the same column of M_{PR}, then we will replace t_k in M_{PR} by t_k / s.

Let M_{PR} be the $m \times m$ weighted place transitive matrix. If a transition t_k appears s times in the same column of M_{PR}, then we will replace t_k by t_k / s in M_{PR}.

It is represented a control flow to a column direction of a transitive matrix. Therefore a control flow can be known through a value of $\sum (\frac{t_k}{s})_i$, where i is a number of places.

Let $\#(pr_i, O(t_k))$ be a number of output value from p_i to transition t_k, and $\#(pr_i, E(t_k))$ be a number of input value from t_k to transition p_i.

A formal definition of the relation conditions in column are as follows.

1) $\#(pr_i, O(t_k)) = \#(pr_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} = 1$: It is maintained a control scope under a firing of the token t_k.

2) $\#(pr_i, O(t_k)) > \#(pr_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} > 1$: It is expanded a control scope under a firing of the token t_k.

3) $\#(pr_i, O(t_k)) < \#(pr_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} < 1$: It is reduced a control scope under a firing of the token t_k.

It is represented a firing possibility of each transition to a row in the transitive matrix. Also, a token exist in the resource common place of a row direction.

And we can know whether a firing possible or not through a value of $\sum \frac{t_{k_i}}{s}$ because the number of transitions is represented a row direction of a resource common place is the number of transitions is inputted at the place. So, if a value of $\sum (\frac{t_{k_i}}{s})_i$ is greater than 1, it is possible to firing.

Let $\#(p_i, O(t_k))$ be a number of output value from p_i to transition t_k, and $\#(p_i, E(t_k))$ be a number of input value from t_k to transition p_i.

A formal definition of the relation conditions in column are as follows.

1) $\#(p_i, O(t_k)) = \#(p_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} \geq 1$

2) $\#(p_i, O(t_k)) > \#(p_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} \geq 1$

3) $\#(p_i, O(t_k)) < \#(p_i, E(t_k)) \Rightarrow \sum \frac{t_{k_i}}{s} \geq 1$

Therefore, we can know that the sum of transitions of a row direction of a transitive matrix is 1 in any case. If a value of $\sum (\frac{t_{k_i}}{s})_i$ is smaller than 1, we can know that the Petri net model mistake.

Let M_{PR} be a transitive matrix of PN = (P,T,I,O,M), then PN is called a deadlock-free if a relationship in row(column) of resource share place $\sum (\frac{t_{k_i}}{s})_i \geq n,$ where n ≥1 and integer, else the PN is deadlock.

Proof: Transitive matrix explains all relationship between the places and transitions by OPN (Ordinary Petri net) in definition. This means that all places have only one input and output arc. So, each place in TM should have been an integer value the number of token. If number of token is less than zero then this place able to reach the deadlock. Also, if number of token is not integer then this place able to reach the deadlock.

3. Deadlock Detection Based on TM

An FMS example introduced in [12,17] is represented in Figure 3.1. There are 3 machines (M1, M2 and M3), one robot and two transport devices, also four operations Job 1, Job 2, Job 3 and Job 4.

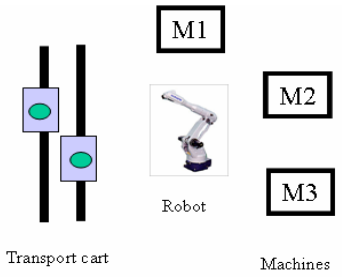

Figure 3.1. System structure

Let's assume that the production ratio is 1/4, which means that the goal is to manufacture 25% of Job1 1, Job 2, Job 3 and Job 4. We define that the incorporate alternative process plans as follows:

Job 1: {M1, M2, M3}

Job 2: {M2,M1,M3}

Job 3: {M1,M2,M3}

Job 4: {M3,M2,M1}

The PN representation of system is as follows:

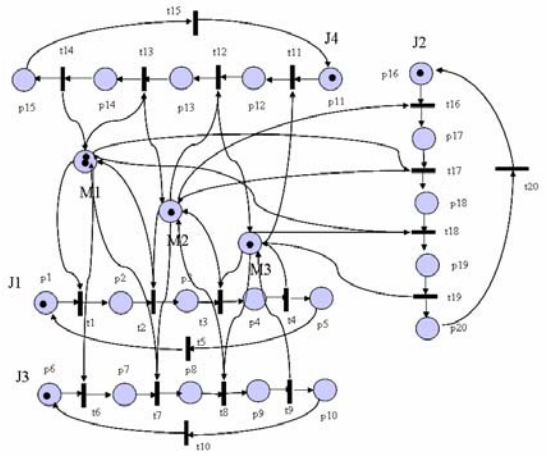

Figure 3.2. Illustrative example: 3 machines and 4 jobs

First, the initial tokens are in M1, M2, M3, p1, p6 and p12. In this case, places p2, p12, p17 have a token and M1,M2 and M3 have not any token for fire. We can say that this marking is dead at M_1.

Based on the avoidance algorithm, we can get an avoidance PN and transitive matrix like as follows:

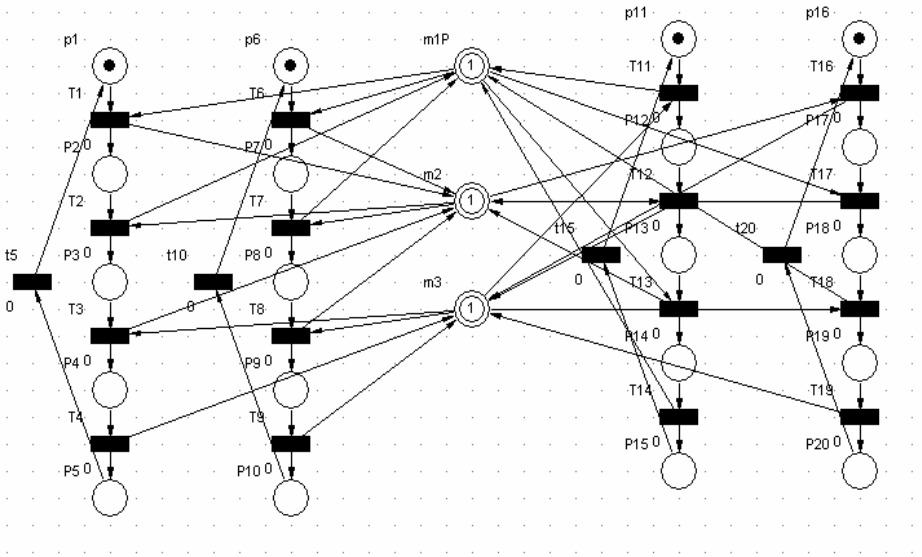

Figure 3.3. Avoidance model of Figure 3.2

Table 3.1. Transitive matrix of Figure 3.3

$$M_{PR}=\begin{bmatrix}
0 & t1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t1/2 & 0 \\
0 & 0 & t2/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t2/2 & 0 & 0 \\
0 & 0 & 0 & t3/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t3/2 & 0 \\
0 & 0 & 0 & 0 & t4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t4 \\
t5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & t6/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t6/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & t7/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t7/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t8/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t8/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t9 \\
0 & 0 & 0 & 0 & 0 & t10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t11/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t11/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t15 & 0 & t12/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t12/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t13/2 & 0 & 0 & 0 & 0 & 0 & 0 & t13/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t14 & 0 & 0 & 0 & 0 & t14/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t16/2 & 0 & 0 & 0 & 0 & t16/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t17/2 & 0 & 0 & t17/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t18/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t19 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t20 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & t1/2 & 0 & 0 & 0 & 0 & t6/2 & 0 & 0 & 0 & 0 & 0 & 0 & t13/2 & 0 & 0 & 0 & t17/2 & 0 & 0 & 0 & 0 \\
0 & 0 & t2/2 & 0 & 0 & 0 & 0 & t7/2 & 0 & 0 & 0 & 0 & t12/2 & 0 & 0 & 0 & t16/2 & 0 & 0 & 0 & t2/2 & t12/2 \\
0 & 0 & 0 & t3/2 & 0 & 0 & 0 & 0 & t8/2 & 0 & 0 & t11/2 & 0 & 0 & 0 & 0 & 0 & 0 & t18/2 & 0 & t11/2 & t3/2 \\
 & & & & & & & & & & & & & & & & & & & & t18/2 & t8/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t7/2 & 0 & t16/2
\end{bmatrix}$$

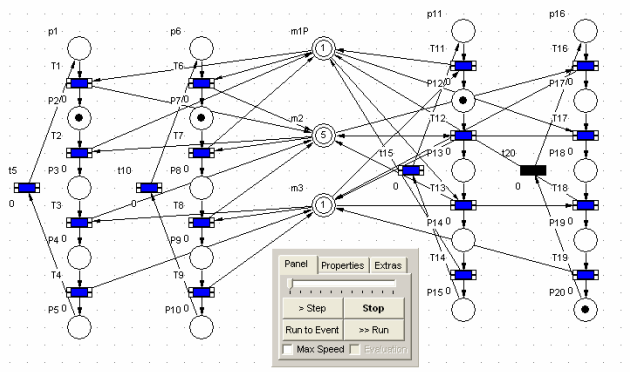

Figure 3.4. Running process of Figure 3.3

In this figure, we can find that modified model has not deadlock status and this model is bounded. Also, we use Visual Object Net ++ (Evaluation Version 1.44.2) to verify the proposed model is not deadlock.

We propose a deadlock detection algorithm based on the previous section. The procedures for the detection and the avoidance of the deadlock status using the resource status table can be illustrated in the following algorithm:

Algorithm: deadlock detection and avoidance

Input: N = <P,T,F,M>

Output: N is deadlock free or not

(1) Define M_{PR} of a Petri net initial N.

(2) Make a marking sequence from the initial place.

(3) Find all relation of resource share places in each column M_{PR}.

(4) Calculate M_R from the initial marking.

(5) If they have one deadlock node then this net N is deadlock net but if not this net is deadlock free.

(6) add one arc to input deadlock status node for another resource share place.

(7) verify the marking sequence with modified M_{PR}.

(8) If this marking sequence accepts fire conditions then this net will be avoided deadlock status and stopped, else this is not deadlock status avoidance, repeat (2) to (6) until satisfy (8).

4. Benchmarking

4.1 Benchmarking model

We introduce an illustrative example module for benchmarking algorithms like as two part types (J_1 and J_2) have to be produced on two machines r_1 and r_2. J_1 contains three operations: r2, r1 and work, J_2 contains three operations: r_1 and r_1 and work. We define that the incorporate alternative process plans as follows:

Job 1: {r2, r1, w}

Job 2: {r1, r2, w}

Figure 4.1 and figure 4. 2 show the Petri nets of this example model:

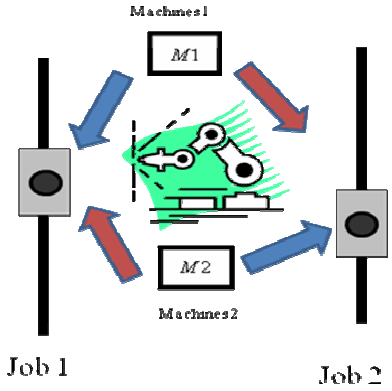

Figure 4.1. Example model for benchmarking

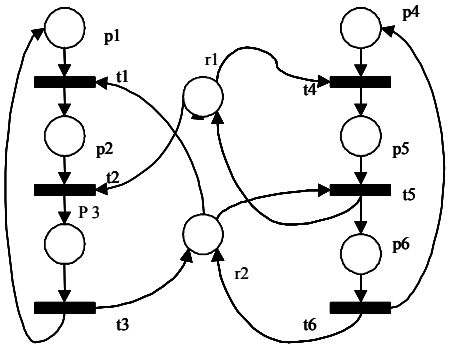

Figure 4.2. Petri Nets of example model

4.2. Deadlock detection algorithms

Three deadlock detection algorithms presented in this section: Siphon, DAPN and the proposed approach(TM) for the FMS example are shown in Figure 4.2. Hence, we consider only deadlock detection algorithms except avoidance case to compare them.

1) Siphon

This algorithm is related with the sequence of the marking in nets. We consider the other usual siphon notations [3,7-8,15,21-22].

A subset of places is called a siphon if $\bullet S \subseteq S\bullet$, ,i.e. any input transition of S is also an output transit ion of S. It is called a trap if $S\bullet \subseteq \bullet S$, A siphon (resp., trap) is said to be minimal if it does not contain other siphons (resp., traps). Also, the following property (see [8]) shows the importance of Siphons and traps in the detection of deadlocks.

Property1: [15]

1) A siphon S free of tokens at a marking remains token- free. Furthermore, all transitions connected to S are not live.

2) A trap marked by a marking remains marked.

3) For any marking such that no transition is enabled, the set of empty places forms a siphon.

In the following, a marking enabled is called dead marking. A siphon S that eventually becomes empty is called potential deadlock [15].

Property2:[15] A Petri net is deadlock-free if no minimal Siphon eventually becomes empty.

In general, deadlock-freeness does not imply liveness. Exceptions include marked graphs and mono-T-semi flow nets.

For some other important classes of Petri nets including FC(Free-choice) Nets and AC(Asymmetric choice) nets, liveness can be still checked by means of siphons and traps[15]. Where, FC net is a Petri net such that $\forall p \in P : |p\bullet| > 1 \Rightarrow \bullet(p\bullet) = \{p\}$ and AC net is a Petri nets such that $\forall p1,p2 \in P: p_1\bullet \cap p_2\bullet \neq \varnothing \; p_1\bullet \subseteq p_2\bullet$ or $p_1\bullet \supseteq p_2\bullet$.

Property3:[15] An FC net is live if every siphon contains a marked trap.

Property4:[15] An ACnet is live if every siphon contains a marked trap.

Property5:[15] siphon which contains a marked trap is not a potential deadlock.

Property6:[15] A Petri net is deadlock-free if for each minimal siphon S either it contains a marked trap or $F(S) > 0$.

Based on these properteis of the siphon, we can find some siphon in example models like as follows:

(1) Siphon:

In this model, we find 4 siphons like as follows:

Siphon	S1	S2	S3	S4
places	{p2,r2,p3}	{r1,p3}	{r2,p2,p3,p6}	{r1,p3,p5}

(2) Analyze the Siphon S1:

S1 is siphon and trap

$\bullet S1 = \{t1,t2,t3\}$

$S1\bullet = \{t1,t2,t3\}$

$\bullet S1 \subseteq S1\bullet,\ S1\bullet \subseteq \bullet S1,$

S2: S2 is Siphon

$\bullet S1 = \{t1,t2,t3\}$

$S1\bullet = \{t1,t2,t3\}$

$\bullet S1 \subseteq S1\bullet,\ S1\bullet \subseteq \bullet S1,$

S3: Siphon and trap

$\bullet S1 = \{t1,t2,t3\}$

$S1\bullet = \{t1,t2,t3\}$

$\bullet S1 \subseteq S1\bullet,\ S1\bullet \subseteq \bullet S1,$

S4: Siphon and trap

$\bullet S1 = \{t1,t2,t3\}$

$S1\bullet = \{t1,t2,t3\}$

$\bullet S1 \subseteq S1\bullet,\ S1\bullet \subseteq \bullet S1,$

Since S1, S3 and S4 are minimal siphons and have a initial marking trap, these subnet are deadlock-free, but S2 has empty marking status, so this subnet is deadlock. The detect algorithm used siphon algorithm is follows:

Input: Petri Nets

Output: siphon, trap

1. Find cycle subnet based on the initial marking
2. Repeat1-2 process due to find all subnet
3. Analyze the siphon and trap at the all subnet
4. Verify the minimal siphon, if find minimal siphon then this net is deadlock, but if not then this net is deadlock-freeness

2) DAPN (Deadlock detection Avoidance Petri Net)

Song et al.[16] proposed an algorithm based on the relational matrix in Petri nets.

In the example model, the concurrence of JOB1 and JOB2 are operated in parallel and the starting points exist in each job, JOB1 and JOB2 can be completed at once. Taking into consideration the deadlock status and concurrent process of JOB1 and JOB2, the matrix is defined as M_J.

$$M_J = \begin{matrix} p_1,p_4 \\ p_2,p_5 \\ p_3,p_6 \end{matrix} \begin{bmatrix} p_1,p_4 & p_2,p_5 & p_3,p_6 \\ 0 & 1_{j1},1_{j2} & 0 \\ 0 & 0 & 1_{j1},1_{j2} \\ 1_{j1},1_{j2} & 0 & 0 \end{bmatrix}$$

The conditions of M_J and the resources sharing places(r1 and r2) must be considered together,

since the interpretation of the deadlock status, its detection, and prevention can be obtained from these. Matching the M_J matrix with the value of the resources sharing places can create the final matrix. The matrix M_{JR} is then obtained in figure 4.4.

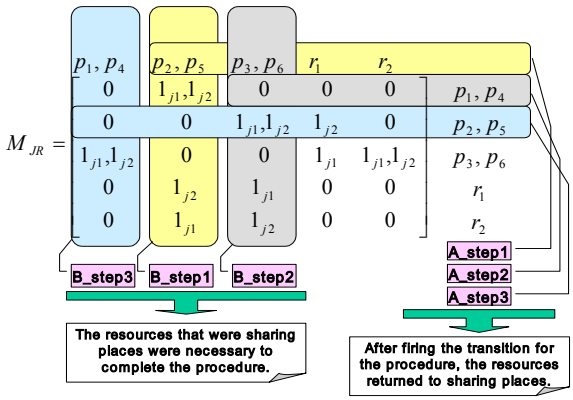

Figure 4.4. Steps for the Algorithm of the example

B_step is a resource stage before the firing transition, which is necessary for job places to proceed to further steps. A_step is a stage of the resources after the firing transition. The job begins by concentrating on the transition from its initially marked location to the next job location. A table is therefore necessary to show the conditions of the resources for job locations in the resources sharing places.

Table. 4.1. Resource State Table of example

Resources Sharing Places	Initial Resources Status	B_step1	A_step1	B_step2	A_step2	B_step3	A_step3
r1	1(j1orj2)	0	0	-1 (j1orj2)	0	0	1(j1orj2)
r2	1(j1orj2)	0	0	-1 (j1orj2)	-1 (j1orj2)	-1 (j1orj2)	1(j1orj2)

The status value of each step's resource can be calculated based on the following steps:

Status value of B_step*i* = Status value of a prior step - Status value of B_step*i*

Status value of A_step*i* = Status value of a prior step + Status value of A_step*i*

A passage in parentheses found next to the resource numbers indicates which job the resource is needed for. For example, 1(j1 or j2) means that this resource can be used for either JOB1 or JOB2. 1(j1) means that this resource can be used only for JOB1.

If the values of r1 or r2 at B_step1 in Table 1 become negative, JOB1 and JOB2 will automatically become negative. Taking into consideration that B_step is before the firing transition, a deadlock status is expected since both jobs got negative results and the necessary resources could not be provided. This part, therefore, becomes the reason for the deadlock status.

3) Transitive matrix (TM)

TM shows all relationship between the places in Petri nets. In this M_{PR}, through definition of TM, we can obtain a relationship of the example model as in the following table M_{PR}'.

Table 4.2. TM of example model

M_{PR}'=

p1	p2	p3	p4	p5	p6	r1	r2		Σ
0	1/2	0	0	0	0	0	0	**P1**	1/2
0	0	1/2	0	0	0	0	0	**P2**	1/2
1	0	0	0	0	0	0	1	**P3**	2
0	0	0	0	1/2	0	0	0	**P4**	1/2
0	0	0	0	0	1/2	1/2	0	**P5**	1
0	0	0	1	0	0	0	1	**P6**	2
0	0	1/2	0	1/2	0	0	0	**r1**	1
0	1/2	0	0	0	1/2	1/2	0	**r2**	3/2
1	1	1	1	1	1	1	2		Σ

In this table M_{PR}, we can find two jobs like as J1: p1-p2-p3 and J2:p4-p5-p6, and this relationship is explained one cycle. Also, we can find some relationship between the resource share places r1 and r2:

Case of r1:

-relation condition in row : 1/2+1/2+r1 = 2

-relation condition in column : 1/2+1/2=1

Case of r2:

-relation condition in row : 1/2+1/2+1/2 = 3/2

-relation condition in column: 1+1=2

In r2, relation condition of row is not equal of 1, this means that the number of token is more less than out and it's may be occurred a deadlock status. In r1, it have all conditions of row and column are satisfied deadlock-free. Finally, this net is deadlock.

By the definition in TM, we can consider a reachable marking as follows:

Mo =[1,0,0,1,0,0,1,1], where [p1,p2,p3,p4,p5,p6,r1,r2].

Then, the reachable marking M_1 is:

$M_1 = Mo \bullet M_{PR}$, then,

M_1 = [0, (p1(t1/2) + r2(t1/2)), r1(t2/2), 0, (p5(t4/2) + r1(t4/2)), r2(t5/2), r2(t5/2), 0].

Since the number of token should be explained by integer value such as equal or greater than zero, we can explain the reachable marking M_1 as follows:

M_1 = [0, 1, 0, 0, 1, 0, 0, 0].

Then next reachable marking is :

$M_2 = M_1 \bullet M_{PR}$

= [0,0, p2(t2/2),0,0,p5(t5/2),0,0]
=[0, 0, 0, 0, 0, 0, 0, 0].

In this result, we can find deadlock status occurred in transitions t2 and t5 by the places r1 and r2.

4.3. Benchmark the algorithms

By the example, we can obtain some results as the following factors (Table 4.3).

1) Understanding

Understanding of the Siphon's algorithm is difficult, because it's difficult to find siphon's subset and analyze the minimal siphon's characteristics. Also, it's some difficult to understand DAPN's algorithm because this algorithm is based on the reachable characteristics. So, it need some effect to find process sequence after made matrix. But, the algorithm of the transitive matrix is consider only a relationship between the resource share places in transitive matrix, so it's some easy to understand the process concept and algorithm.

2) Complexity

The complexity of siphon algorithm is $O(n^2)$ in the example model. Also, in DAPN, the complexity is $O(n^4)$ and in TM the complexity is $O(n^2)$. This solution showed that the good complexity of siphon and proposed algorithm rather than DAPN.

3) Time

Base on three algorithms, we can get the best time result for the good solution. It shows 32 sec for Siphon algorithm, 72 for DAPN's and 18 for proposed algorithm (TM). This means that the proposed approach analyzing time is shorter than the Siphon and DAPN's approaches.

Finally, the result of relation graph of three approaches is shown in Figure 4.6. In this figure, we can find the optimization is TM's is good, and Siphon's approach is effectiveness in the time.

Table 4.3. Benchmark results

	Siphon	DAPN	TM
complexity	$O(n^2)$	$O(n^4)$	$O(n^2)$
time	32	72	18
understanding	difficult	More difficult	easy

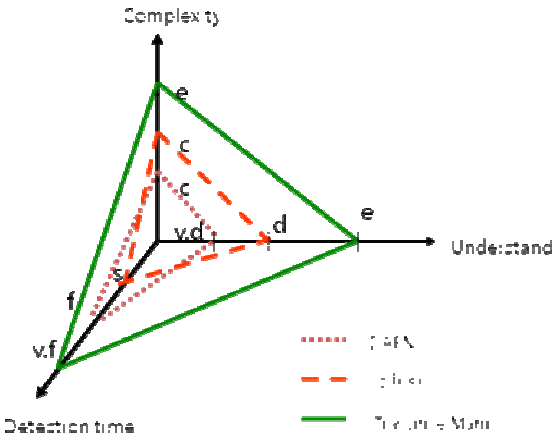

Figure 4.6. Total relational graph

5. Conclusion

In this paper, we focused on the avoidance policy of the deadlock problem in Petri nets using the transitive matrix. The transitive matrix explained all relationship between the places and transitions in the net. By these relationships, we can find deadlock status based on the relation between the resource share places. Finally, it can be said that it is easy to find the deadlock status and modify the deadlock status in the net. In this work, a simple example has been shown to compare three algorithms: siphon, DAPN and the proposed approach. After applying to a system with two machines and two jobs, we compared the results based on the three factors like as time, complexity and understandings. These results indicated that our method could be an easily understandable tool to find deadlock and to compute feasibility time.

Acknowledgement

We thank the LT 2007 staffs for recommending this work to the editors of SIC, as Prof. Niculescu and Prof. Jung for valuable comments and suggestions.

REFERENCES

1. CORBETT, J.C., **Evaluating Deadlock detection methods for concurrent software**, IEEE tr. Sof. Eng. Vol. 22(3), 1996, pp. 161-180.
2. DAMASCENO, B.C. and XIE X, **Petri nets and deadlock-free scheduling of multiple-resource operations**, In: IEEE SMC'99, 1999, pp. 878-883.
3. EZPLETA, J., COLOM J.M., MARTINEZ J., **A Petri net based deadlock prevention policy for flexible manufacturing systems**, IEEE tr. Robotics and Automat., Vol. 11, No. 2, 1995, pp. 173-184.
4. LIU, J., ITOH Y., MIYAZAWA I., SEIKIGUCHI T., **A Research on Petri nets Properties using Transitive matrix**, In: Proceeding IEEE SMC99, 1999, pp. 888-893.
5. LEE, J., KORBAA O., **Scheduling analysis of FMS: An unfolding time Petri nets approach, mathematics and Computer simulation**, 70, 2006, pp. 419-432.
6. MELZER, S. and ROMER S., **Deadlock checking using net Unfoldings**, In: Proc. of the Conf. on Computer-Aided Verfication (CAV'97), 1997.

7. MURATA, T., **Petri Nets: Properties, Analysis an Applications**, Proceedings of the IEEE, 77(4), IEEE, USA, 1989, pp. 541-580.

8. PETERSON, J.L., **Petri Net Theory and the Modeling of Systems**, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981.

9. SHATZ, S.M., TU S., MURATA T., **An application of Petri net reduction for Ada Tasking Deadlock Analysis**, In: IEEE Trans. on Parallel and Distributed Systems, Vol. 7, No. 12, 1996, pp. 1307-1322.

10. XIONG, H.H., ZHOU M.C., **Deadlock free scheduling of an automated manufacturing system based on Petri nets**, In: IEEE ICRA'97, 1997, pp. 945-950.

11. LEE, J., **Deadlock find algorithm using the Transitive Matrix**, In: Proceeding CIE'04, 2004.

12. A. GIUA, et elc., observer-Based state-feedback control of timed Petri nets with deadlock recovery, In: IEEE Trans. on Automatic control, Vol. 49, No. 1, 2004, pp. 17-29.

13. KE-YI, X., HU BS, CHEN HX, **Deadlock Avoidance Policy for Petri net Modeling of Flexible Manufacturing Systems with Shared resources**, In: IEEE Trans. on Automatic control, Vol. 41, No. 2, 1996, pp. 289-295.

14. BASIL, F., A. GIUA, C. SEATZU, **Observer-based state-feedback control of time Petri nets with deadlock recovery: theory and implementation**, In: Proceed. Of Symp. On Discrete events in Industrial and manufacturing systems, CESA 2003, 2003.

15. CHU, F., X. XIE, **Deadlock Analysis of Petri Nets using siphons and mathematical programming,** In: IEEE Trans on Robotica and Automation, Vol.13, No. 6, 1997, pp. 793-804.

16. YUJIN, SONG and JONGKUN LEE, **The study on the Deadlock Avoidance using the DAPN and the Adjacency Matrix**, In Journal of the Korea Society for simulation, vol. 15, no. 3, 2006, pp. 1-10.

17. KIM, S.-H., S-H LEE, J-K. LEE, **Deadlock analysis of Petri nets based on the Resource Share Places Relationship**, In Studies in Informatics and control, vol. 16, no. 1, 2007, pp. 33-44.

18. KIM, S.-H., J-K. LEE, **Benchmark Study of Deadlock analysis methods in FMS**, In: proceeding of LT2007, Tunis, 2007, pp. 363-368.

19. PARK, J., S. A. REVELIOTIS, **Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible manufacturing**, In: IEEE tr. Robotics and Automat., Vol. 46, 2001, pp. 1572-1583.

20. DOTOLI, M., M.P. FANTI, **Deadlock Detection and Avoidance strategies for Automated Storage and Retrieval Systems**, In: IEEE Trans. on SMC-C, Vol. 37, No. 4, 2007, pp. 541-552.

21. LI, ZW, MC ZHOU, **Elementary Siphons of Petri Nets and Their Application to deadlock prevention in Flexible manufacturing Systems**, In: IEEE Trans. on SMC-A, Vol. 34, No. 1, 2004, pp. 38-51.

22. ABDALLAH, I.B., H.A. ELMARAGHY, **Deadlock Prevention and Avoidance in FMS: A Petri Net Based Approach** , In: J. Adv. Manuf. Technol., Vol. 14, 1998, pp. 704-715.

23. GIRAULT, C., R. VALK, **Petri Nets for Systems Engineering**, Springer-Verlag, Berlin Heidelberg, 2003.