Deadlock Avoidance of a Kind of JSP with Multi-resources Sharing *

Jing Li, Hejiao Huang , Farooq Ahmad
Dept. of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China

{hjhuang@hitsz.edu.cn; lijing@hitsz.edu.cn; farooq190@ hotmail.com}

Abstract

This paper presents the scheduling problem with multi-resource sharing, which each operation may need more than one kinds of resource. Timed Petri net is used to formulate this problem to analyze deadlock and minimize the makespan. A deadlock avoidance policy addressed here consists of three stages: The first stage is deadlock detection based on transitive matrix and circular waiting. The second one is deadlock recovery by adding new arcs to destroy deadlock. The last stage is deadlock-free design. The efficiency of the method proposed in this paper is illustrated by an example in the end.

Keywords: Petri net, JSP, Deadlock Avoidance, Transitive matrix

1. Introduction

Job-shop scheduling problems (JSP) can be stated as follows: there are a finite set of *n* jobs, each job which consists of a chain of operations, and a finite set of *m* machines and some other resources, such as workers and AGVs, which are able to process at most one operation at a time. Each operation requires one or more than one resource, but at most one unit of each type of resource. Each resource can process at most one operation at a time, and each operation must be processed during an uninterrupted time on some given resources. The purpose is to find a schedule, that is, a reasonable allocation of the operations to time intervals to machines or other resources such that the makespan is minimal. The allocation of resources has to be considered for the modeling and functioning of JSP systems, which is a challenge for a system designer, especially in multi-resource sharing system.

Petri nets have been extensively used to model and analyze discrete event systems including JSP [1]-[4]. It is useful for preventing and avoiding unexpected system behaviors, such as deadlock and capacity overflow. Siphon are mainly technology on deadlock analysis [3]-[4]. Some scheduling algorithms are also used to get a feasible scheduling, such as heuristic scheduling algorithm and dynamic programming approach [5]-[6].

Transitive matrix of Petri net is another strategy to analyze deadlock [7]. The entries of place transitive matrix describe the transferring relation from one place to another place through transitions. Deadlock structures, deadlock avoidance method are given in [8]-[9]. The theorems above have been extensively studied to

* This work was financially supported by National Natural Science Foundation of China with Grant No. 10701030.

prevent deadlock and analyze the properties of Petri nets. However, the compositional system that shares common resources could not grab much attention of the researchers, especially for multi-resources sharing system.

In this paper, timed Petri net is addressed to model the JSP with multi-resources sharing. Deadlock conditions are analyzed by place transitive matrix and a deadlock recovery method is introduced to construct a deadlock-free system.

This paper is organized as follow. The next section introduces system modeling by timed Petri net. In Section 3, some basic knowledge about deadlock is presented. Deadlock avoidance is stated in Section 4 based on deadlock detection and recovery. Section 5 describes a case study about JSP. Some conclusive remarks are presented in section 6.

2. Timed Petri Net Modeling

The notations used in this paper are mostly taken from [1], [9]-[10].

Definition 1 A *Timed Petri net* (*TPN*) is a *6-tuple* $(N, M_0) = (P, T, I, O, M_0, \tau)$, where P is a finite set of places; T is a finite set of transitions and $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$; I: $T \rightarrow P^{\infty}$ is the input function, a set of directed arcs from places to transitions, $I(t_j)$ is the set of input place of t_j; O: $T \rightarrow P^{\infty}$ is the output function, a set of directed arcs from transitions to places, $O(t_j)$ is the set of output place of t_j; M_0 is the initial marking. τ is a time function τ: $T \rightarrow N$ (N is the set of positive integers), which is associated to transitions such that $\forall t \in T$, $\tau(t) = n$ represents the firing time of transition t.

Definition 2 $\forall p_i \in P$ and $\forall t_j \in T$, A place p_i is an input place of transition t_j if $p_i \in I(t_j)$ and p_i is an output place of transition t_j, if $p_i \in O(t_j)$. Denote the weight of directed arc (p_i, t_j) as $\#(p_i, I(t_j))$, and the weight of directed arc of (t_j, p_i) as $\#(p_i, O(t_j))$.

Definition 3 $B^-[i, j]$ is the matrix of input function with m rows and n columns, where $B^-[i, j] = \#(p_i, I(t_j))$; $B^+[i, j]$ is the matrix of output function with m rows and n columns, where $B^+[i, j] = \#(p_i, O(t_j))$. $B = B^+ - B^-$ is called *incidence matrix*.

Definition 4 Let L_P be *place transitive matrix* and let L_V be *transition transitive matrix* with m rows and m columns, where $L_P = B^-(B^+)^T$ and $L_V = (B^+)^T B^-$. The *labeled place transitive matrix* L_{VP} is a $m \times m$ matrix satisfying: $L_{VP} = B^- \, diag(t_1, t_2, \ldots, t_n) \, (B^+)^T$, where the elements of L_{VP} describe the direct transferring relation from one place to another through one or more transitions. The *Weighted place transitive matrix* is denoted as L_{VP}^*, If a transition t_k appears s times in the same column of L_{VP}, then replace t_k in L_{VP} by t_k/s in L_{VP}^*, otherwise the elements in L_{VP}^* are the same as L_{VP}.

Definition 5 Consider two Petri nets $(N_1, M_{10}) = (P_1, T_1, I_1, O_1, M_{10})$ and $(N_2, M_{20}) = (P_2, T_2, I_2, O_2, M_{20})$, where $P_1 \cap P_2 = R \neq \varnothing$. Let $(N, M_0) = (P, T, I, O, M_0)$ be composed from (N_1, M_{10}) and (N_2, M_{20}) by operation *place fusion*. Then the elements in (N, M_0) are defined as follows: $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $I = I_1 \cup I_2$, $O = O_1 \cup O_2$ and

$$M_0(p) = \begin{cases} M_{10}(p) & p \in P_1 \\ M_{20}(p) & p \in P_2 \\ \max\{M_{10}(p), M_{20}(p) & p \in R \end{cases}$$

Each job will be a small Petri net with a unique path of transitions and places. Each operation in a job corresponds to a transition t, and the input (output) place of transition t is an operation place P that can be interpreted as the start (end) of the operation. The resource places are connected to the transitions which operations require these resources. Moreover, for each resource occupied by some operation t, when it is not used by

the next operation, this resource can be released after operation t finished.

When the operations are ready to implement or the resources are available, there will be tokens in those places. Moreover, the original (final) state of each job corresponds to the initial marking M_0 (final marking M_f) of the Petri net model. The whole model is got by common place merging, and this n jobs' scheduling problem is converted to find a firing sequence in the resultant Petri net model to get a deadlock-free system in order to calculate the makespan.

3. Deadlock Analysis

Some definitions about deadlock used in next section are presented here.

3.1. Some Notations about Deadlock

A transition is said to be a *dead transition* at marking M if M is not the final marking M_f and there is no reachable marking to make transition t enable. Formally, transition t is a dead transition: $\exists\ M \in R\ (N, M_0) \wedge (M \neq M_f)$, $\forall t \in T : \urcorner M[t>$, then.

Since our modeling method is based on the composite designing, the "circular waiting" discussed in this paper is between two tasks. In manufacturing system, "*circular waiting*" is said to be a state that for two tasks, each of them hold a machine but both of them are waiting for the second machine occupied by the other. It is one condition of deadlock in manufacturing system.

In Petri net, "*circular waiting*" can be demonstrated as, for Petri net N_i (i =1, 2), r_1 and r_2 are resource places in N_i and t_{ia} t_{ib} t_{ic} t_{id} (i =1, 2) is sub-transition path which is associated with resource places, where N_1 satisfies: r_1 is associated with the pairs of transitions (t_{1a}, t_{1c}) and (t_{1b}, t_{1d}), and $r_1\bullet = t_{1a}$, $\bullet r_1 = t_{1c}$ and $r_2\bullet = t_{1b}$, $\bullet r_2 = t_{1d}$; N_2 satisfies: r_1 is associated with the pairs of transitions (t_{2a}, t_{2c}) and (t_{2b}, t_{2d}), and $r_1\bullet = t_{2b}$, $\bullet r_1 = t_{2d}$ and $r_2\bullet = t_{2a}$, $\bullet r_2 = t_{2c}$.

Deadlock occurs when there are dead transitions that cause "*circular waiting*" in the system. A net is said to be *deadlock-free* if and only if there is no deadlock in the system.

3.2. Transformation of Marking

In this sub-section, the notation about transformation equation [7]-[9] and the improved form on JSP are presented here. The *transformation of marking* is defined as $M_R(k+1)^T = M(k)^T L_{VP}^*$, where $M_R(k+1)$ is a m-vector of nonnegative integer and is a reachable marking from $M(k) = [p_1(k), p_2(k), \ldots, p_m(k)]^T$, L_{VP}^* is the weighted place transitive matrix [9]. At a state $M(k)$, If t_i fires, we define $| t_i | = 1$; and if there is a token in p_i at $M(k)$,define $| p_i(k) | = 1$.

The transformation equation shows the flow relation of a token in $p_i(k)$ from $M(k)$ to $M_R(k+1)$ by the weighted place transitive matrix. In this equation, if $\sum_i p_i L_{VP_{ij}}^*$ is integer, then let $\sum_i p_i L_{VP_{ij}}^* = 1$, which means $p_i(k)$ can be transferred from $M(k)$ to $M_R(k+1)$; otherwise, if $\sum_i p_i L_{VP_{ij}}^*$ is not integer, let $\sum_i p_i L_{VP_{ij}}^* = 0$, which means $p_i(k)$ can not be transferred from $M(k)$ to $M_R(k+1)$ [9].

However, $M_R(k+1)$ only presents the flow relation of tokens, it may not necessary correspond to $M(k+1)$. In order to improve this situation, we add some constrained conditions to this equation:

(1) When $\sum_i p_i L_{VP_{ij}}^*$ is integer, e.g. $\sum_i p_i L_{VP_{ij}}^* = 1$ in $M(k+1)$, then token can be transferred from $M(k)$ to $M(k+1)$;

(2) If $\sum_i p_i L_{VP_{ij}}^*$ is not integer, $\sum_i p_i L_{VP_{ij}}^* = 0$ in $M(k+1)$, then token can't be transferred;

(3) The token of place p_i in condition (2) which is not the same as place in condition (1) is not changed;

(4) The token in other place is the same as $M_R(k+1)$.

From this calculation we can find out a series of reachable marking from M_0. In addition, if there is no reachable marking at M_i, the next reachable marking calculated will be the same as M_i. Thus we can find out whether a marking is reachable or not from M_0.

4. Deadlock Avoidance Method

Deadlock detecting algorithm is addressed to find out deadlock in the compositional system by merging two small Petri nets with shared resources.

Algorithm I: Deadlock Detection

Input: $N_1 = (P_1 \cup R, T_1, F_1, M_{10}, \tau_1)$, $N_2 = (P_2 \cup R, T_2, F_2, M_{20}, \tau_2)$, the Initial marking M_{10}, M_{20} and the final marking M_{1f}, M_{2f}.

Output: Deadlock or not

Step 1: Do place merging on N_i and N_j, and then generate a net N with the initial marking M_0 and the finial marking M_f.

Step 2: Construct the weighted place transitive matrix-L_{VP}^* for compositional net N.

Step 3: For $k = 0$, $k < n$, k ++

Calculate the next marking $M_R(k+1)$ from the marking $M_R(k)$ by $M_R(k+1)^T = M(k)^T L_{VP}^*$

If M_R is equal to M_f, stop, output "There is no deadlock in the net, wecan get the makespan".

else if there are dead transitions and these dead transitions construct a "*circular waiting*" structure, then output "There is a deadlock, we should do deadlock recovery on the model".

else output "There are mistakes in the system model".

If deadlock occurs, the system falls into global deadlock and stops running. Therefore, deadlock recovery is important in system design. Since deadlock is caused by "*circular waiting*", if we destroy the "*circular waiting*" structure, deadlock will disappear and the system will be deadlock-free. Therefore, a recovery method is proposed using "self-loop" to destroy "*circular waiting*" structure in this subsection.

Consider, N_1 and N_2 are two Petri net models with shard places r_1 and r_2. They satisfy the "circular waiting" conditions mentioned in section 3.1. Suppose $\sigma_i = t_{i0}\ t_{i1}\ \ldots t_{im}$ is a transition path in N_i, $\sigma_i' = t_{ia}\ t_{ib}\ t_{ic}\ t_{id}$ is a sub-transition path of σ_i (i = {1, 2}), and # ($t_{i0}\ t_{i1}\ \ldots t_{i(a-1)}$) is the makespan of $t_{i0}\ t_{i1}\ \ldots t_{i(a-1)}$.

Algorithm II: Deadlock Recovery

Input: Dead transitions which cause deadlock, suppose they are t_{1b} and $t_{1b'}$.

Output: A recovery model

Step 1: Compare # ($t_{10}\, t_{11}\, \ldots\, t_{1(a-1)}$) and # ($t_{20}\, t_{21}\, \ldots\, t_{2(a-1)}$) and find out the greater quantity, without loss the generality, suppose it is # ($t_{10}\, t_{11}\, \ldots\, t_{1(a-1)}$).

Step 2: Add two new arcs $t_{1(b-1)} \rightarrow r_1$ and $r_1 \rightarrow t_{1b}$ into the model, then the pair (t_{1a}, t_{1c}) is decomposed as (t_{1a}, $t_{1(b-1)}$) and (t_{1b}, t_{1c}) which are both associating with r_1.

Step 3: Deadlock detection by algorithm I.

If output is "There is a deadlock", do the recovery similarly as Step1 and Step2.

else, there is no deadlock, stop.

After the recovery given above, the sub-system becomes deadlock-free. The method proposed above only change the input and output of a resource place in one sub-Petri net. If the complexity of the model is not considered, we can do the similar change in the second sub-Petri net, and the Petri net model may be live and the makespan will be minimized at a time.

4.1. Deadlock-free Method

The deadlock detection and recovery algorithm are mainly for a compositional system with two small Petri nets sharing common resources. For an n jobs Petri

nets system, we do deadlock detection and recovery on each pair of Petri nets. Then do place merging on these *n* Petri nets and a deadlock-free modeling is got. The makespan of the scheduling can be calculated by simulations on CPN-Tool.

5. A Case Study

An example with two jobs and two kinds of resources is shown in Table 2 in order to describe the technology of deadlock detection and recovery.

Table 2: A Job Shop Scheduling Problem

	O_1	O_2	O_3	O_4
J_1	M_1: 7	M_1, M_2: 4	M_1, M_2: 3	M_2: 4
J_2	M_2: 3	M_1, M_2: 5	M_1, M_2: 4	M_1: 3

In order to get the scheduling, we solve this problem gradually:

Step 1: System model

Fig.2 depicts the Petri nets for two jobs, and t_{i1}, t_{i2}, t_{i3}, t_{i4} are corresponding to the four operations in job i (i = {1, 2}), r_1 and r_2 are corresponding to the two resources. The initial marking of job i is M_{i0} = (1, 0, 0, 0, 0, 1, 1) ^{T} and the final marking is M_{if} = (0, 0, 0, 0, 1, 1, 1) ^{T}.

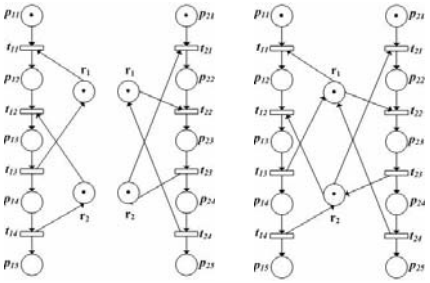

Fig.2. Petri nets models Fig.3. Place fusion

Step 2: Deadlock detection

- Do place merging on the common resources places r_1 and r_2 :

Fig.3 is the compositional system for this problem, the initial marking of the merging system M_0 = (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1) ^{T} and the final marking M_f = (0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1) ^{T}.

- Construct the weighted place transitive matrix-L_{VP}^{*} :

Table 3 is the weighted place transitive matrix of Fig.3, $M(k)$=[$p_{11}(k)$, $p_{12}(k)$, $p_{13}(k)$, $p_{14}(k)$, $p_{15}(k)$, $p_{21}(k)$, $p_{22}(k)$, $p_{23}(k)$, $p_{24}(k)$, $p_{25}(k)$, $r_1(k)$, $r_2(k)$]^{T} and the initial marking is M_0 = (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1) ^{T}.

Table 3: L_{VP}^{*} of Petri net in Fig.3

$$L_{VP}^{*} = \begin{array}{c} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{15} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{25} \\ r_1 \\ r_2 \end{array} \begin{bmatrix} 0 & t_{11}/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{12}/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_{13} & 0 & 0 & 0 & 0 & 0 & t_{13} & 0 \\ 0 & 0 & 0 & 0 & t_{14} & 0 & 0 & 0 & 0 & 0 & t_{14} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & t_{21}/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{22}/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{23} & 0 & t_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{24} & t_{24} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{11}/2 & 0 & 0 & 0 & 0 & 0 & t_{22}/2 & 0 & 0 & 0 \\ 0 & 0 & t_{12}/2 & 0 & 0 & 0 & t_{21}/2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate the reachable marking by transformation of marking: $M_R(k+1)^T = M(k)^T L_{VP}^{*}$:

The next reachable marking is:

$M_1 = M_0 L_{VP}^{*}$ = [0, p_{11} ($t_{11}/2$) + r_1 ($t_{11}/2$), r_2 ($t_{12}/2$), 0, 0, 0, p_{21} ($t_{21}/2$) + r_2 ($t_{21}/2$), r_1 ($t_{22}/2$), 0, 0, 0, 0]^{T} = [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]^{T}, then

$M_2 = M_1 L_{VP}^{*}$ = [0, 0, p_{12} ($t_{12}/2$), 0, 0, 0, 0, p_{22} ($t_{22}/2$), 0, 0, 0, 0]^{T} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^{T}

In M_2, transitions t_{12} and t_{22} are dead transitions connected to resource places r_1 and r_2, which involve "circular waiting". Then the system is confronted a deadlock.

Step 3: Deadlock recovery

- Since (# t_{11} = 7)>(# t_{21} = 3), the first sub-model will be modified;

Add two arcs t_{11} → r_1 and r_1 → t_{12} to get a recovery model in Fig.4.;

- Deadlock detection again and there is no deadlock, stop.

When there are more than two jobs, deadlock detection and recovery should be implemented in each two jobs to make sure each sub-system deadlock-free. Then a deadlock-free compositional model can be built by place fusion.

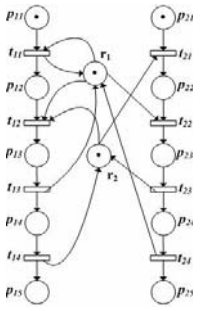

Fig.4.The recovery mode

Step 4: Scheduling resolution

Based on the algorithms above, the system is deadlock-free. The makespan of this JSP is 30, and the Gantt chart is shown in Fig.5.

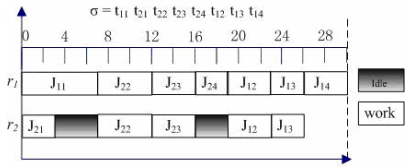

Fig.5. Gunter chart of the scheduling

6. Conclusion

In this paper, deadlock avoidance method in JSP with multi- resources sharing is considered by timed Petri nets. Place transitive matrix is applied to detect deadlock. A deadlock recovery strategy is introduced based on "self-loop". However, the work focuses on deadlock caused by two tasks. Our future work will do some deeply research in a more complex system, which deadlock is caused by more than three tasks with more kinds of shared resources.

7. Reference

[1] J. L. Peterson, "Petri Net Theory and the Modeling of System", Prentice Hall, (1981).

[2] H.J. Huang, L. Jiao and T.Y. Cheung, "Property-preserving subnet reductions for designing manufacturing systems with shared resources", Theoretical Computer Science, 332(1-3): 461-485(2005).

[3] Hesuan Hu, Zhiwu Li and Anrong Wang, "On the optimal set of elementary siphons in Petri nets for deadlock control in FMS", 2006 IEEE International Conference of Networking, Sensing and Control, pp. 244-247(2006).

[4] Zhiwu Li and Na Wei, "Deadlock control of flexible manufacturing systems via invariant-controlled elementary siphons of Petri nets", The International Journal of Advanced Manufacturing Technology, Vol. 33, pp. 24-35(2007).

[5] Xu Gang and Wu Zhiming, "A kind of deadlock-free scheduling method based on Petri net", IEEE HASR'02, pp. 195-200(2002).

[6] Kim Y., Tatsuya T., and Taysuo N., "FMS scheduling based on timed Petri net model and reactive graph search", Applied Mathematical Modelling, Vol.31, pp.955-970(2007).

[7] Liu J., Itoh Y., Miyazawa I. and Sekiguchi T., "A research on Petri net properties using transitive matrix", Proceeding IEEE SMC99, pp. 888-893(1999).

[8] Yujin. Song and Jongkun. LEE, "Deadlock analysis of Petri nets using the transitive matrix", SICE Annual Conference 2002, Osaka, (2002).

[9] Sanghwan Kim, Sangho Lee and Jongkun Lee, "Deadlock analysis of Petri nets nased on the reousrce share places relationship", IMACS multiconference on CESA, pp. 59-64(2006).

[10] Murata T, "Petri net: properties, analysis, and applications", Proc IEEE, Vol. 77, pp. 541-580(1985).