# A Simple Petri Net Controller by Solving Some Integer Linear Programming Problems

**A. Dideban[1], M. Zareiee[2], A. A. Orouji[3]**

*[1,3]Electrical and Computer Eng. Department, Semnan University, Semnan, Iran, Tel: 98-231-3354123 ; e-mail[1]: adideban@semnan.ac.ir; e-mail[3]: aliaorouji@ieee.org.*
*[2] Scool of Engineering, Damghan University, Damghan, Iran, e-mail[2]: mzareiee@du.ac.ir.*

**Abstract:** The number of control places for designing a simple Petri Net controller is important. So, many efforts have been accomplished during last decade to design a controller with small number of such places. But, the number after applying some of these methods is still large and some of the other methods are complicated. In this paper, we have attempted to develop the previous methods for obtaining a simple controller in a systematic way. In this method, a small number of control places are obtained by solving a few numbers of Integer Linear Programming Problems at which the numbers of constraints and variables in each problem smoothly grow with respect to the numbers of reachable states. Also, the obtained controller model is maximally permissive.

*Keywords:* Supervisory control, Discrete Event System, Petri Net, Control place, Integer Linear Programming Problem.

## 1. INTRODUCTION

Supervisory control theory in Discrete Event Systems (DES) tries to prevent the system from entering the forbidden states by restricting its behaviour (Ramadge and Wonham, 1987, 1989). This restriction is performed by disabling the controllable events in special conditions. The forbidden states are the ones that violate specification or are deadlock states. For evaluating DESs, Petri Net (PN) can be applied as a suitable tool to model these systems (Krogh and Holloway, 1991). In recent years, a lot of methods based on PN models have been proposed for avoiding the forbidden states.

(Ghaffari et al., 2003) have presented a method based on regions theory to achieve desired performance. In this method, some constraints are generated which some of them verify the authorized states and the others are for violating the forbidden states. Then, by solving some Integer Linear Programming (ILP) problems, some control places can be calculated that adding them to the system leads to obtaining a maximally permissive controller. But, the drawback of this method is its computational complexity. Moreover, a lot of control places is generated which causes a complicated model.

In flexible manufacturing systems (FMS), deadlock is a highly undesirable situation and the system should be prevented from entering them (Abdallah and ElMaraghy, 1998; Huang, 2007; Hu and Li, 2009; Piroddi et al., 2009). There are a lot of methods for avoiding the deadlock states. In these methods, control places are calculated using Siphon theory. But, the number of such places is large and some of them may be redundant. So, the redundant control places can be removed (Uzam et al., 2007; Li and Hu, 2009).

Restricting the weight sum of tokens in some places can lead to verifying the specifications or avoiding the forbidden states. The restriction can be performed using some constraints which are called Generalized Mutual Exclusion Constraints (GMEC). (Giua et al., 1992) have proposed a method for assigning GMECs to forbidden states in safe PNs. This method has been developed by (Chen et al., 2011) and in our previous work (Zareiee et al., 2011) at which the GMECs can be obtained in non safe PNs. Enforcing GMECs on the system is performed using control places (Yamalidou et al., 1996). However, the numbers of GMECs and control places are large when the number of forbidden states is large.

(Dideban and Alla, 2005) have proposed a method for reducing the number of GMECs in safe and conservative PNs. This method simplifies the GMECs using invariant property. But, for this simplification, a large state space should be verified. This method is developed in (Dideban and Alla, 2008) at which some covering markings of forbidden states are prevented and a small number of GMECs are obtained. The method in (Dideban et al., 2009) develops the last method where verifying the small state space leads to simplification of the GMECs. However, all the mentioned methods can be applied to safe PNs. The method proposed by (Zareiee et al., 2014) can reduce the number of GMECs in non safe petri nets but this number is still large.

In our previous work (Zareiee and Dideban, 2011), we have presented a method for obtaining a GMEC for avoiding all the forbidden states in non safe PNs by solving an ILP problem. But, this method can generate an answer in small systems. (Chen and Li, 2011) have proposed a useful method for obtaining the minimal number of control places by solving an ILP problem. However, the problem of this

method is its computational complexity which makes it inapplicable to large scale PNs.

In this paper, we want to develop the ideas in (Zareiee and Dideban, 2011; Chen and Li, 2011) and propose a new method for obtaining a small number of GMECs. So, a few ILP problems with small numbers of constraints and variables are solved. For this reason, at first step, some constraints are generated which some of them verify the authorized states (these constraints are called safe constraints) and the others violate the forbidden states (these constraints are called unsafe constraints). Then, by classifying the unsafe constraints, some ILP problems are solved which their solutions cause the following results:

(a) Each one of the safe constraints is verified by all the solutions of the ILP problems which this leads to verifying all the authorized states by the obtained GMECs.

(b) Each one of the unsafe constraints is verified by at least one of the solutions of the ILP problems. This leads to avoiding all the forbidden states.

(c) A small number of control places are obtained by solving a few number of ILP problems with small numbers of constraints and variables.

The rest of this paper is as follows. In section 2, the important definitions and basic concepts are presented. The new method for reducing the number of control places is proposed in section 3. This method is changed in section 4 where a controller with small numbers of control places, arcs, and tokens are obtained. Finally, conclusions are presented in section 5.

## 2. BASIC CONCEPTS

In this section, the basic concepts and definitions are introduced which are important for presenting the new methods. Also, we suppose that the reader is familiar with the PNs basis (David and Alla, 2005) and supervisory control theory (Ramadge and Wonham, 1987, 1989).

A PN structure is represented by a quadruplet $R = (P, T, W, M_0)$, where $P$ is the set of places, $T$ is the set of transitions, $W$ is the incidence matrix and $M_0$ is the initial marking. Places and transitions are connected together by arcs and the relation between them is stated by the incidence matrix. Places can be marked by tokens and are divided into two types: safe places and non safe places. If all places in a PN model are safe, this PN model is called safe PN. In safe PN, the number of tokens in each place is one or zero. But, in non safe PN, this number can be more than one. The marking of a PN is a column vector where $i^{th}$ component is the marking of place $p_i$. For the sake of simplicity, we write the markings in the transposed form as follows:

$$[m_1\ m_2\ \dots\ m_n]^T$$

where $m_i$ is the number of tokens in place $p_i$ and $n$ is the number of places.

In a PN, the set of all reachable markings (or reachable states) is denoted as $\mathcal{M}_R$. $\mathcal{M}_R$ is divided into two sets of authorized states ($\mathcal{M}_A$) and forbidden states ($\mathcal{M}_F$). The set of forbidden states $\mathcal{M}_F$ is defined as follows:

- The set of states that do not respect the specification which is denoted as $\mathcal{M}_b$.
- Deadlock states or the set of states from which the system reaches the deadlock states inevitably and are shown by $\mathcal{M}_d$.
- The set of states from which the system can uncontrollably reach the marking set $M_b \cup M_d$.

The set of reachable states without $\mathcal{M}_F$ corresponds to the set of authorized states.

In the set of forbidden states, there is an important subset which is called the set of border forbidden states and is defined in Definition 1.

**Definition 1:** $\mathcal{M}_B$ is the set of border forbidden states and is defined as follows:

$$\mathcal{M}_B = \{M_i \in \mathcal{M}_F \mid \text{if } \exists\, M_j \in \mathcal{M}_A, M_j \xrightarrow{t_i} M_i \ \Rightarrow\ t_i \in T_c\}$$

where $T_c$ is the set of controllable transitions. ❑

The border forbidden states are the ones that preventing them leads to preventing all the forbidden states. So, for obtaining the maximally permissive controller, it is enough to forbid these states. The whole explanations about the border forbidden states are explained by (Kumar and Holloway, 1996).

### *2.1 GMECs and enforcing them on the system using control places*

GMECs are the constraints that limit the weight sum of tokens in some places. Enforcing GMECs on the system may comply the specification or prevent the system from entering the forbidden states. In safe PNs, these constraints can be constructed easily. However, in non safe PNs, constructing GMECs is difficult. The GMECs in safe PNs can be constructed as the following form:

In a safe PN, suppose that when the places $p_{i1}, p_{i2}, \dots, p_{in}$ are marked, there is a forbidden state. So, the GMEC related to this state is constructed as follows:

$$\sum_{i=1}^{n} m_{ik} \le n-1$$

where $m_{ik}$ is the number of tokens in place $p_{ik}$ and $n$ is the number of marked places (Giua et al., 1992).

The method for constructing GMECs in non safe PNs is described in Algorithm 1.

**Algorithm 1 (Zareiee et al., 2011):**

**Input:** The set of authorized states $\mathcal{M}_A=\{[z_{11}\ z_{12}\ \dots\ z_{1n}],\ \dots,\ [z_{r1}\ z_{r2}\ \dots\ z_{rn}]\}$ and a forbidden state $M_1=[z_{B1}\ z_{B2}\ \dots\ z_{Bn}]$, where $r$ is the number of authorized states and $n$ is the number of places.

**Output:** A GMEC related to $M_1$.

***Step 1.*** Consider a generic constraint as follows:

$$k_1m_1+k_2m_2+\dots+k_nm_n \le x \qquad (1)$$

where $x$ and $k_i$ for $i$=1, 2 ,…, $n$ are non negative integers and $m_i$ is the number of tokens in place $p_i$.

***Step 2.*** Substitute the markings of all the authorized states in the constraint (1) and construct the following constraints:

$$[z_{11}\ z_{12}\ \dots\ z_{1n}] \rightarrow k_1z_{11}+k_2z_{12}+\dots+k_nz_{1n} \le x \qquad (2\text{-}1)$$

. .

. .

. .

$$[z_{r1}\ z_{r2}\ \dots\ z_{rn}] \rightarrow k_1z_{r1}+k_2z_{r2}+\dots+k_nz_{rn} \le x \qquad (2\text{-}r)$$

***Step 3.*** Substitute the marking of $M_1$ in the constraint (1) and convert the smaller equal sign to greater sign as follows:

$$[z_{B1}\ z_{B2}\ \dots\ z_{Bn}] \rightarrow k_1z_{B1}+k_2z_{B2}+\dots+k_nz_{Bn} > x \qquad (3)$$

***Step 4.*** Solve the set of the relations (2-1) to (2-$r$) and (3) which is an ILP problem and obtain the minimum values of $x$ and $k_i$ for $i$=1, 2, …, $n$. (in this problem the objective function is: minimum($k_1+k_2+\dots+k_n+x$) where $x>0$ and $k_i \ge 0$ for $i$=1,2,…,$n$).

***Step 5.*** Substitute $x$ and $k_i$ (for $i$ = 1, 2, …, $n$) obtained from step 4, in the constraint (1). The resultant constraint is a GMEC for the forbidden state $M_1=[z_{B1}\ z_{B2}\ \dots\ z_{Bn}]$. ❑

By using Algorithm 1, constructing GMECs for forbidden states is possible. GMECs can be enforced on the system using control places (Yamalidou et al., 1996). In this case, for each GMEC, a control place is added to the system. For obtaining such places, suppose that the set of GMECs is shown as follows:

$$L.M_P^T \le b$$

where $M_P$ is the marking vector ($M_P=[m_1\ m_2\ \dots\ m_n]$), $L$ is a $n_c\times n$ matrix, $b$ is a $n_c\times 1$ vector, $n_c$ is the number of GMECs and $n$ is the number of places. The elements in $L$ and $b$ are non negative integers. As it was mentioned, for a GMEC, a control place is connected to the PN model. So, for each GMEC, a row is added to the incidence matrix of the system. These rows are calculated as follows:

$$W_c=-L.W_P$$

where $W_P$ is the incidence matrix of system before connecting the control places. Therefore, the incidence matrix of the system after connecting the control places is as follows:

$$W=\begin{bmatrix} W_P \\ W_c \end{bmatrix}$$

The initial marking vector of the control places is calculated as the following form:

$$M_{c0}{}^T=b-L.M_{P0}{}^T$$

where $M_{P0}$ is the initial marking of the system before connecting the control places. The initial marking vector of the system after connecting the control places is:

$$M_0^T=\begin{bmatrix} M_{P0}^T \\ M_{c0}^T \end{bmatrix}$$

### *2.2 Reducing the space of states that should be verified or forbidden by the controller*

The set of places in a PN model of an FMS is classified into three groups: Idle, Operation and Resource places. To calculate the set of control places for preventing the system from entering the deadlock states, the markings of operation places are only considered (Uzam and Zhou, 2006). So, the number of states that should be verified and the ones that should be forbidden by the controller are reduced (Chen et al., 2011). To explain this concept, the over-state and covering state concepts are defined as follows:

**Definition 2:** Suppose that $M_1$ and $M_2$ are two marking vectors in a PN model where $M_1\le M_2$. In this case, $M_1$ is an over-state of $M_2$ and also $M_2$ is a covering state of $M_1$. ❑

By using this definition, two theorems are introduced to see how it is possible to reduce the numbers of states which should be authorized or forbidden.

**Theorem 1 (Dideban and Alla, 2008):** Suppose that $M_1$ and $M_2$ are two forbidden states where $M_1\le M_2$. If $M_1$ is forbidden by a GMEC, $M_2$ is forbidden by this GMEC. ❑

**Theorem 2 (Chen et al., 2011):** Suppose that $M_1$ and $M_2$ are two authorized states where $M_1\le M_2$. If $M_2$ is not forbidden by a GMEC, $M_1$ is not forbidden by this GMEC. ❑

As it was mentioned, the markings of operation places should be only considered to calculate the control places of an FMS. So, the markings of idle and resource places can be eliminated from the set of reachable markings. After this elimination, in the set of forbidden states, some states may be the over-states of the other ones. In this case, according to Theorem 1, it is enough to prevent the over-states for avoiding all the forbidden states. Therefore, the set of states that should be forbidden is reduced. The set of these over-states is shown with $M_{O\text{-}F}$. Moreover, some authorized states may be covering states of the other authorized states. According to Theorem 2, for verifying all the authorized states by the controller, it is enough to verify the covering states of authorized states. Thus, the number of states that should be verified is reduced. The set of covering states of authorized states is shown with $M_{C\text{-}A}$. So, the calculation for

obtaining GMECs becomes simpler by finding the sets of $M_{O\text{-}F}$ and $M_{C\text{-}A}$.

## 3. A NEW METHOD FOR OBTAINING A SMALL NUMBER OF CONTROL PLACES

Algorithm 1 is a helpful way for assigning a GMEC to each forbidden state. However, when the number of these states is large, a large number of GMECs are generated which lead to connecting a large number of control places. We have somewhat solved this problem in our previous work (Zareiee and Dideban, 2011) at which by solving an ILP problem, one GMEC is obtained. In this method, we consider all the forbidden states in step 3 of Algorithm 1. It means that all their markings are substituted in constraint (1) and the smaller equal signs are converted to greater signs. Then, an ILP problem composed of the constraints in steps 2 and 3 is solved. However, this method is only applicable to small systems and cannot generate any answer for large systems. (Chen and Li, 2011) have proposed an effective method for obtaining a small number of GMECs by solving an ILP problem. The main drawback of this method is its computational complexity which makes it inapplicable to large scale PN models. In this section, we want to develop the methods in (Zareiee and Dideban, 2011; Chen and Li, 2011) and propose an effective method for obtaining a small number of control places by solving a few number of ILP problems. Moreover, the numbers of variables and constraints in each ILP problem are small. This new method is introduced in Algorithm 2.

**Algorithm 2:**

**Input:** The set of authorized states $\mathcal{M}_A = \{[z_{11}\, z_{12}\, \ldots\, z_{1n}], \ldots, [z_{r1}\, z_{r2}\, \ldots\, z_{rn}]\}$ and the set of border forbidden states $\mathcal{M}_B = \{[B_{11}\, B_{12}\, \ldots\, B_{1n}], \ldots, [B_{t1}\, B_{t2}\, \ldots\, B_{yn}]\}$.

**Output:** A small number of GMECs.

**Default:** $t = 0$, $R_t = \varnothing$, $W_t = \varnothing$, $W_t^t = \varnothing$, $R_t^t = \varnothing$

***Step 1.*** Consider a generic constraint as follows:

$$k_1 m_1 + k_2 m_2 + \ldots + k_n \le x \tag{4}$$

where $m_i$ is the number of tokens in place $p_i$.

***Step 2.*** Substitute the markings of the authorized states in the constraint (4) and consider the obtained constraints as follows:

$$\sum_{i=1}^{n} z_{j,i}.k_i \le x \qquad j = 1, 2, ..., r \tag{5}$$

which are called safe constraints.

***Step 3.*** Substitute the markings of the border forbidden states in the constraint (4) and convert the smaller equal sign to greater sign and consider the obtained constraints as the following form:

$$\sum_{i=1}^{n} B_{l,i}.k_i > x \qquad l = 1, 2, ..., y \tag{6}$$

which are called unsafe constraints. The set of unsafe constraints is denoted as $SS_g$.

***Step 4.*** $t = t+1$

***Step 5.*** Solve the first ILP problem in section 3.1 (the relations (7) to (10)), and obtain the constants $x$ and $k_i$ (for $i$= 1, …, $n$) which verify all the safe constraints and the largest number of unsafe constraints (this step is described in subsection 3.1).

***Step 6.*** Save the obtained constants in the set $W_t$ and then remove the verified unsafe constraints from the set of unsafe constraints ($SS_g$) and substitute them in the set $R_t$.

***Step 7.*** If $SS_g$ is not empty, go to step 4.

***Step 8.*** For $q$ = 2, 3, …, $t$, perform the following operations:

- $R_q \cup$ (the set of safe constraints) = $V_q$
- $(R_1 \cup R_2 \cup \ldots \cup R_{q-1}) = Z_q$
- Solve the second ILP problem in section 3.1 (the relations (11) to (15)), and obtain the constants $x$ and $k_i$ (for $i$=1, …, $n$) which verify all the constraints in the set $V_q$ and the largest number of constraints in the set $Z_q$ (this step is described in section 3.1).
- Substitute the obtained constants in the set $W_q^q$ and then consider the verified constraints of $Z_q$ in the set $R_q^q$. Add the constraints of $R_q$ to $R_q^q$.

***Step 9.*** Choose the smallest number of the sets of constants $x$ and $k_i$ (for $i$=1, …, $n$) among the sets $W_q$ and $W_q^q$(for $q$ = 1, 2, …, $t$) which in sum verify all the unsafe constraints in step 3 (this step is described in subsection 3.2).

***Step 10.*** Substitute the final sets of constants in the constraint (4). These constraints are the small number of GMECs which enforcing them on the system leads to obtaining a maximally permissive controller). ❑

By using Algorithm 2, a small number of control places can be obtained where connecting them to the system, leads to a maximally permissive controller. Moreover, some ILP problems are solved and the numbers of constraints and variables in each problem are small with respect to size of model.

### *3.1 Finding the constants x and $k_i$ (for i=1, …, $N_i$) for verifying the largest number of unsafe constraints*

In this section, we want to introduce a method for finding the constants $x$ and $k_i$ which verify all safe constraints and the largest number of unsafe constraints (step 5 of algorithm 2). To do this, an ILP problem is considered as follows:

$$\min F = \sum_{l \in N_{SSg}} f_l \tag{7}$$

$$\sum_{i=1}^{n} z_{j,i}.k_i - x \leq 0 \qquad j = 1,2,...,r \tag{8}$$

$$\sum_{i=1}^{n} B_{l,i}.k_i - x > -Q \times f_l \qquad \forall l \,|\, l \in N_{SSg} \tag{9}$$

$$f_l \in \{0,1\} \tag{10}$$

where $Q$ is a positive constant which should be considered large enough and $N_{SSg}$ denotes $\{l \,|\, (\sum_{i=1}^{n} B_{l,i}.k_i > x) \in SS_g\}$. In the above ILP problem, $f_l$ is related to $l^{th}$ unsafe constraint. So, $f_l$ = 0 means that $l^{th}$ unsafe constraint is verified by the obtained constants $x$ and $k_i$, and $f_l$=1 means that $l^{th}$ unsafe constraint is not verified by these constants. Solving this ILP problem generates the constants $x$ and $k_i$ verifying the largest number of the unsafe constraints and all the safe constraints.

Now, to find the constants $x$ and $k_i$ for verifying all the constraints in $V_q$ and the largest number of constraints in $Z_q$, we consider the following ILP problem (step 8 of Algorithm 2):

$$\min F = \sum_{l \in (N_R^1 \cup N_R^2 \cup ... \cup N_R^{q-1})} f_l \tag{11}$$

$$\sum_{i=1}^{n} z_{j,i}.k_i - x \leq 0 \qquad j = 1,2,...,r \tag{12}$$

$$\sum_{i=1}^{n} B_{l,i}.k_i - x > 0 \qquad \forall l \in N_R^q \tag{13}$$

$$\sum_{i=1}^{n} B_{l,i}.k_i - x > -Q \times f_l \qquad \forall l \in (N_R^1 \cup N_R^2 \cup ... \cup N_R^{q-1}) \tag{14}$$

$$f_l \in \{0,1\} \tag{15}$$

where $N_R{}^q$ denotes $\{l \,|\, (\sum_{i=1}^{n} B_{l,i}.k_i > x) \in R_q\}$.

By solving this ILP problem, some unsafe constraints which were eliminated from $SS_g$ (and exist in the sets of $R_d$ for $d<q$) are added to the set $R_q$.

### *3.2 Final selection in the sets of the obtained constants x and $k_i$ (for i=1, ..., n)*

In the sets of the obtained constants $x$ and $k_i$ (for $i$=1, …, $n$), there may be a set which verifies some unsafe constraints or there may be the same unsafe constraint which is verified by some sets of constants (it means that there may be an unsafe constraint which is common between the sets of $R_t$ (for $t$ = 1, 2, …)). The method for selecting the final sets of the constants is similar to the final selection in Quine-McCluskey method for simplifying logical expressions (Morris Mano, 2001). This final selection can be performed as the following form:

If there is an unsafe constraint which is only verified by one set of constants, this set should be selected. Then, all the unsafe constraints which are verified by this set should be eliminated. Then, in the set of residual unsafe constraints, if an unsafe constraint is verified by two or several sets of constants, it is necessary to choose the set which verifies the most numbers of non-selected unsafe constraints. In the case of equality, the simplest set should be selected.

Now, after introducing this method, an example is considered to show the capability of Algorithm 2.

**Example 1.** Consider the Resource Allocation System (RAS) taken from (Reveliotis and Choi, 2006). The PN model of this system is illustrated in Fig. 1. This model consists of three resource types, $R_1$, $R_2$ and $R_3$ and two processes. The first and the second processes are modelled by the paths "$t_{10}P_{11}t_{11}P_{12}t_{12}P_{13}t_{13}$" and "$t_{20}P_{21}t_{21}P_{22}t_{22}P_{23}t_{23}$", respectively. The places $P_{10}$ and $P_{20}$ are characterized as the idle places for the first and second processes respectively. The initial marking of each one of these places establishes the upper bound to the number of instances of each processes that can be simultaneously loaded into the system. In this system, all the transitions are controllable and the system should be prevented from entering the deadlock states.

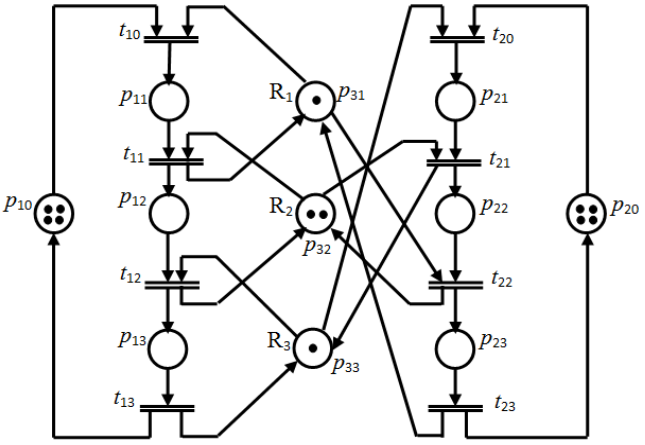


Fig. 1. The resource allocation system taken from (Reveliotis and Choi, 2006).

In this system, there are 47 states at which 42 ones are authorized and 5 ones are forbidden. The forbidden states (which are also border forbidden states) are the ones that preventing them leads to preventing the system from entering the deadlock states. In this example, the numbers of states in $M_{C\text{-}A}$ and $M_{O\text{-}F}$ are 6 and 3, respectively.

By using Algorithm 2, two solutions are obtained. So, the first solution is as follows:

$k_{11} = k_{22} = 2$, $k_{12} = k_{21} = 1$, $k_{13} = k_{23} = 0$, $x = 5$.

Therefore, the GMEC related to this solution is:

$$2m_{11}+m_{12}+m_{21}+2m_{22} \leq 5 \quad (16)$$

And the second solution is as the following form:

$k_{12} = k_{21} = 1$, $k_{13} = k_{23} = k_{11} = k_{22} = 0$, $x = 2$.

The GMEC related to this solution is:

$$m_{12}+m_{21} \leq 2 \quad (17)$$

The incidence matrix related to these two GMECs is as follows:

$$W_c = \begin{bmatrix} -2 & 1 & 1 & 0 & -1 & -1 & 2 & 0 \\ 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 \end{bmatrix} \quad (18)$$

And the initial marking vector of the control places is:

$$M_{c0} = \begin{bmatrix} 5 \\ 2 \end{bmatrix} \quad (19)$$

Connecting the two control places to the system leads to obtaining a maximally permissive controller.

## 4. A NEW METHOD FOR OBTAINING A SIMPLE CONTROLLER WITH SMALL NUMBERS OF CONTROL PLACES, ARCS AND TOKENS

The proposed method in Algorithm 2 is very good for obtaining a small number of control places. But, the numbers of generated arcs and tokens may not be the least numbers. So, in this section, by considering some changes in Algorithm 2, another method is introduced to obtain a small numbers of control places, arcs and tokens. To see this concept, Example 2 is considered.

**Example 2.** Consider the FMS in Fig. 2 taken from (Uzam, 2002). The sets of idle, resource and operation places are $P^0=\{p_1, p_8\}$, $P_R=\{p_{14}, ..., p_{19}\}$ and $P_A=\{p_2, ..., p_7, p_9, ..., p_{13}\}$, respectively. It has 282 reachable states at which 205 ones are authorized and 77 ones are forbidden states. The numbers of states in $M_{C\text{-}A}$ and $M_{O\text{-}F}$ are 26 and 8, respectively.

By applying Algorithm 2, two GMECs are obtained as follows:

$$m_2+2m_3+m_4+2m_5+2m_6+3m_9+3m_{10} \leq 9 \quad (20)$$

$$4m_2+8m_3+4m_4+5m_5+m_9+m_{10}+8m_{11}+7m_{12} \leq 14 \quad (21)$$

So, the incidence matrix and the initial markings are as the following form:

$$W_c = \begin{bmatrix} -1 & -1 & 0 & -1 & 0 & 0 & 2 & 0 & -3 & 0 & 3 & 0 & 0 & 0 \\ -4 & -4 & 0 & -1 & 3 & 5 & 0 & 0 & -1 & 0 & -7 & 1 & 7 & 0 \end{bmatrix} \quad (22)$$

$$M_{c0} = \begin{bmatrix} 9 \\ 14 \end{bmatrix} \quad (23)$$

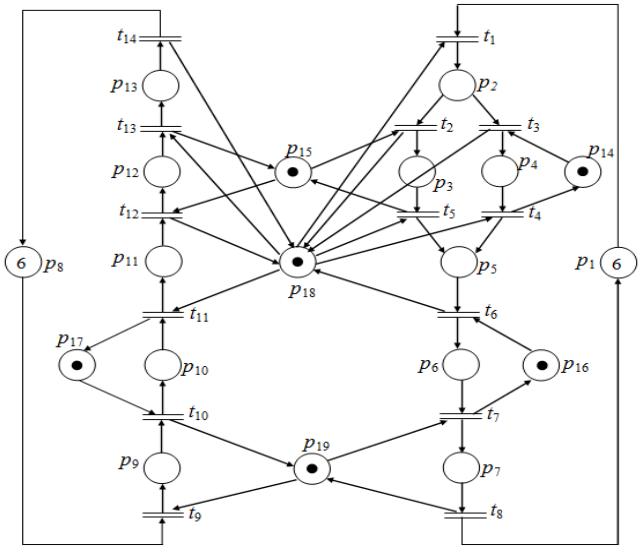


Fig. 2. A Felexible manufacturing system with 282 reachable states.

As it is obvious, by using Algorithm 2, two control places are obtained that enforcing them on the system leads to obtaining a maximally permissive controller. But, the numbers of arcs and tokens can be reduced. For instance, if we add the constraint $x \leq 13$ to the ILP problems in Algorithm 2, the following GMECs can be obtained:

$$m_2+2m_3+m_4+2m_5+2m_6+3m_9+3m_{10} \leq 9 \quad (24)$$

$$m_2+2m_3+m_4+2m_{11}+2m_{12} \leq 3 \quad (25)$$

where their incidence matrix and initial tokens are calculated as follows:

$$W_c = \begin{bmatrix} -1 & -1 & 0 & -1 & 0 & 0 & 2 & 0 & -3 & 0 & 3 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 \end{bmatrix} \quad (26)$$

$$M_{c0} = \begin{bmatrix} 9 \\ 3 \end{bmatrix} \quad (27)$$

So, by adding the constraint $x \leq 13$ to the ILP problems in Algorithm 2, the numbers of arcs and tokens are reduced. Now, we explain how it is possible to add such constraints to the ILP problems. For this reason, we can firstly apply Algorithm 2 and save the obtained solutions. Then, we should consider the biggest $x$ among the answers as $x_g$ and apply Algorithm 2 again by adding the constraint $x<x_g$ to the ILP problems in Section 3.1. After that, we save the obtained solutions. Again, we consider the biggest $x$ among the solutions in this step as $x_g$ and apply Algorithm 2 by considering $x<x_g$ and save the obtained solutions. We perform this until $x_g$ is one or Algorithm 2 does not produce a solution. At the end, a set of GMECs is obtained. So, in this set, the smallest number of GMECs should be selected as the ones that enforcing them on the system leads to obtaining a maximally permissive controller. In the case of equality, the GMEC with smallest $x$ should be selected. This method is described in Algorithm 3.

**Algorithm 3:**

**Input:** The set of authorized states $\mathcal{M}_A = \{[z_{11}\ z_{12}\ \dots\ z_{1n}], \dots, [z_{r1}\ z_{r2}\ \dots\ z_{rn}]\}$ and the set of border forbidden states $\mathcal{M}_B = \{[B_{11}\ B_{12}\ \dots\ B_{1n}], \dots, [B_{t1}\ B_{t2}\ \dots\ B_{yn}]\}$.

**Output:** A small number of GMECs.

**Default:** $t = 0$, $R_t = R_t^t = \varnothing$, $W_t = W_t^t = \varnothing$, $V_q = \varnothing$, $Z_q = \varnothing$ $SS_g = \varnothing$, $SS_{se} = \varnothing$, $U_G = \varnothing$

***Step1.*** Apply Algorithm 2 and substitute the obtained GMECs from Algorithm 2 in the set $U_G$.

***Step2.*** Select the biggest $x$ ($x$ is the number in the right side of each GMEC) among all the GMECs in $U_G$ and consider it as $x_g$, and $t = 0$.

***Step 3.*** Consider a generic constraint as follows:

$$k_1m_1+k_2m_2+\dots+k_n \le x \tag{28}$$

where $m_i$ is the number of tokens in place $p_i$.

***Step 4.*** Substitute the markings of the authorized states in the constraint (28) and consider the obtained constraints as follows:

$$\sum_{i=1}^{n} z_{j,i}.k_i \le x \qquad j = 1,2,...,r \tag{29}$$

which are called safe constraints. Substitute them in the set $S_{se}$.

***Step 5.*** Substitute the markings of the forbidden states in constraint (28) and convert the smaller equal sign to greater sign and consider the obtained constraints as the following form:

$$\sum_{i=1}^{n} B_{l,i}.k_i > x \qquad l = 1,2,...,y \tag{30}$$

which are called unsafe constraints. Substitute them in the set $S_g$.

***Step 6.*** If $x_g$ is not greater than 1, go to Step 19.

***Step 7.*** $SS_g = S_g$ and $SS_{se} = S_{se}$

***Step 8.*** $t = t+1$

***Step 9.*** Add the constraint $x \le x_g$ to the first ILP problem in section 3.1 (the relations (7) to (10)). Solve the ILP problem and obtain the constants $x$ and $k_i$ (for $i = 1, \dots, n$) which verify all the safe constraints in the set $SS_{se}$ and the largest number of unsafe constraints in $SS_g$ (this step is described in subsection 3.1. But, we should add the constraint $x < x_g$ to the ILP problems in this subsection).

***Step 10.*** *If* $f_1 = f_2 = \dots = 1$, $t > 1$

*Then*

substitute the constants of the sets $W_1$, $W_2$, …, $W_{t-1}$ in the constraint (28) and add the obtained GMECs to the set $U_G$.

go to Step 19,

*Else*,

*If* $f_1 = f_2 = \dots = 1$, $t = 1$

the set $W_q^q$ and then add the verified constraints of $Z_q$ to the set $R_q^q$. Add the constraints of $R_q$ to $R_q^q$.

***Step 13.*** Choose the smallest number of the sets of constants $x$ and $k_i$ (for $i = 1, \dots, n$) among the sets $W_q$ and $W_q^q$ (for $q = 1, 2, \dots, t$) which in sum verify all the unsafe constraints (this step is described in subsection 3.2).

***Step 14.*** Substitute the final sets of constants in the constraint (28).

***Step 15.*** Select the biggest $x$ among the obtained $x$ in Step 13 and consider it as $x_g$.

***Step 16.*** Add the obtained GMECs to the Set $U_G$.

***Step 17.*** $W_1 = W_2 = \dots = W_t = R_1 = R_2 = \dots R_t = \varnothing$ and

$W_1^1 = W_2^2 = \dots = W_t^t = R_1^1 = R_2^2 = \dots R_t^t = \varnothing$ and $t = 0$

***Step 18.*** Go to Step 6.

***Step 19.*** Choose the smallest number of GMECs from the set $U_G$ which are violated by all the border forbidden states (similar to the method which is described in section 3.2. More information about this selection is described in (Dideban and Alla, 2008; Dideban et al., 2009).

***Step 20.*** *End.* ❑

*go to step 19*

*Else,*

save the obtained constants in the set $W_t$ and then remove the verified unsafe constraints from the set $SS_g$ and substitute them in the set $R_t$.

***Step 11.*** If the set $SS_g$ is not empty, go to step 8.

***Step 12.*** For $q = 2, 3, \dots, t$, perform the following operations:

- $R_q \cup$ (the set of safe constraints) $= V_q$
- $(R_1 \cup R_2 \cup \dots \cup R_{q-1}) = Z_q$
- Add the constraint $x \le x_g$ to the second ILP problem in section 3.1 (the relations (11) to (15)). Solve the ILP problem and obtain the constants $x$ and $k_i$ (for $i = 1, \dots, n$) which verify all the constraints in the set $V_q$ and the largest number of constraints in the set $Z_q$ (this step is described in subsection 3.1. But we should add the constraint $x < x_g$ to the ILP problems in this subsection).
- If there is a solution (it means if at least one of $f_i$s is equal to zero), substitute the obtained constants in

**Table 1. Performance comparison of the new methods with a conventional method**

| *Examples* → | Example 1 | | | | | | | Example 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Methods* ↓ | $N_{C\text{-}A}$ | $N_{O\text{-}F}$ | $N_{cp}$ | $N_{arc}$ | $N_{ILP}$ | $N_c$ | $N_v$ | $N_{C\text{-}A}$ | $N_{O\text{-}F}$ | $N_{cp}$ | $N_{arc}$ | $N_{ILP}$ | $N_c$ | $N_v$ |
| **Algorithm 2** | 6 | 3 | 2 | 10 | 3 | 9, 7 | 10, 8 | 26 | 8 | 2 | 15 | 3 | 34, 30 | 20, 16 |
| **Algorithm 3** | 6 | 3 | 2 | 10 | 9 | 7, 8, 9, 10 | 8, 9, 10 | 26 | 8 | 2 | 12 | 25 | 27, 28, 29, 30, 31, 34, 35 | 13, 14, 15, 16, 17, 20 |
| **Chen and Li (2011)** | 6 | 3 | 2 | 10 | 1 | 33 | 27 | 26 | 8 | 2 | 12 | 1 | 328 | 152 |

By applying Algorithm 3 to Example 2, two GMECs are obtained the same as relations (24) and (25) where their incidence matrix and initial tokens are in (26) and (27), respectively. As it is obvious from (24) and (25), the numbers of arcs and initial tokens have reduced compared to applying Algorithm 2. Algorithm 3 introduces a good method for obtaining a small numbers of control places, arcs and tokens. But it should solve some ILP problems more than Algorithm 2, and Algorithm 2 is simpler.

## 5. DISCUSSION

In this paper, two methods in algorithms 2 and 3 are proposed where using them, a small number of control places can be obtained. These two methods can obtain the same number of control places. But, their difference is related to the number of their arcs and also the computational complexity. Algorithm 2 is a useful method that can obtain an acceptable solution by solving a small number of ILP problems. Also, the numbers of constraints and variables in each ILP problem are small. However, the number of arcs is not the least number. Without considering the number of arcs, the method in Algorithm 2 is very good; otherwise, Algorithm 3 can generate a small numbers of control places and arcs. Compared to Algorithm 2, Algorithm 3 should solve more ILP problems. There is a trade off between Algorithms 2 and 3. So, by considering the numbers of arcs, Algorithm 3 is better than Algorithm 2 and by considering computational complexity, Algorithm 2 is better. The computational complexity of Algorithm 3 is more than Algorithm 2 and we use Algorithm 3 when the number of arcs is important otherwise Algorithm 2 is more efficient than Algorithm 3.

By considering Examples 1 and 2 in Table 1, the new methods are compared to the proposed method by Chen and Li (2011). In this table, $N_{C\text{-}A}$, $N_{O\text{-}F}$, $N_{cp}$, $N_{arc}$, $N_{ILP}$, $N_c$ and $N_v$ are the numbers of covering states of authorized states, over-states of forbidden states, control places, arcs, ILP problems that should be solved, constraints and variables, respectively. As it is obvious from Table 1, after applying the proposed method by (Chen and Li, 2011), the numbers of constraints and variables exponentially grow with respect to the size of model ($N_{C\text{-}A}$ and $N_{O\text{-}F}$) which this concept makes it inapplicable to large scale PNs. By using Algorithms 2 and 3, the numbers of constraints and variables smoothly grow with respect to the size of model. In these algorithms, more ILP problems should be solved, but, the numbers of constraints and variables in each problem are small. In fact, instead of using an ILP problem with large numbers of constraints and variables, some ILP problems with small numbers of constrains and variables are solved.

## 6. CONCLUSIONS

This paper presents two methods for avoiding the forbidden states and designing a maximally permissive controller with small number of control places. So, some integer linear programming problems should be solved. The problem of the proposed methods is the computational complexity for generation of reachable markings. The number of reachable markings increases exponentially with respect to the size of model. However, the numbers of constraints and variables in the integer linear programming problems are not polynomial with respect to the numbers of elements in the set of authorized states and forbidden states which is an advantage of the proposed methods.

## REFERENCES

Abdallah, I.B., and ElMaraghy, H.A. (1998). Deadlock prevention and avoidance in FMS: a petri net based approach, *International Journal of Advanced Manufacturing Technology*, 14(10), 704-715.

Chen, Y., and Li, Z.W. (2011). Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems, *Automatica*, 47(5), 1028-1034.

Chen, Y., Li, Z.W., Khalgui, M., and Mosbahi, O. (2011). Design of a Maximally Permissive Liveness-Enforcing Petri Net Supervisor for Flexible Manufacturing Systems, *IEEE Transactions on automation science and engineering*, 8(2), 374-393.

David, R., and Alla, H. (2005). *Discrete, continuous, and hybrid Petri Nets*, Springer.

Dideban, A., and Alla, H. (2005). From forbidden states to linear constraints for the optimal supervisory control, *Control Engineering and applied Informatics (CEAI)*, 7(3), 48-55.

Dideban, A., and Alla, H. (2008). Reduction of Constraints for Controller Synthesis based on Safe Petri Nets, *Automatica*, 44(7), 1697-1706.

Dideban, A., Zareiee, M., and Alla, H. (2013). Controller Synthesis with Highly Simplified Linear Constraints, *Asian Journal of Control*, 15(1), 80-94.

Ghaffari, A., Rezg, N., and Xie, X.L. (2003). Design of live and maximally permissive Petri Net controller using the theory of regions, *IEEE Transactions on Robotics and Automation*, 19(1), 137-142.

Giua, A., DiCesare, F.M., and Silva, M. (1992). Generalized Mutual Exclusion Constraints on Nets with Uncontrollable Transitions, *In Proc. IEEE int. conf. on systems, man, and cybernetics*, 974–799.

Hu, H., and Li, Z. (2009). Efficient deadlock prevention policy in automated manufacturing systems using exhausted resources, *International Journal of Advanced Manufacturing Technology*, 40, 566-571.

Huang, Y.S. (2007). Design of deadlock prevention supervisors using Petri nets, *International Journal of Advanced Manufacturing Technology*, 35, 349-362.

Krogh, B.H., and Holloway, L.E. (1991). Synthesis of feedback control logic for discrete manufacturing systems, *Automatica*, 27(4), 641–651.

Kumar, R., and Holloway, L.E. (1996). Supervisory control of deterministic Petri nets with regular specification languages, *IEEE Transactions on Automatic Control*, 41(2), 245-249.

Li, Z., and Hu, H. (2009). On systematic methods to remove redundant monitors from liveness-enforcing net supervisors, *Computer & Industrial Engineering*, 56(1), 53-62.

Morris Mano, M. (2001). *Digital design*, Prentice Hall.

Piroddi, L., Cossalter, M., and Ferrarini, L. (2009). A resource decoupling approach for deadlock prevention in FMS, *International Journal of Advanced Manufacturing Technology*, 40, 157-170.

Ramadge, P.J., and Wonham, W.M. (1987). Modular feedback logic for discrete event systems, *SIAM Journal of Control and Optimization*, 25(1), 1202-1218.

Ramadge, P.J., and Wonham, W.M. (1989). The control of discrete event systems, Dynamics of discrete event systems [Special issue]. *Proceedings of the IEEE*, 77(1), 81-98.

Reveliotis, S.A., and Choi, J.Y. (2006). Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri Nets through the theory of regions, *LNCS*, 4024, 322-341.

Uzam, M. (2002). An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions, *International Journal of Advanced Manufacturing Technology*, 19(3), 192– 208.

Uzam, M., Li, Z., and Zhou, M. (2007). Identification and elimination of redundant control places in Petri net based liveness enforcing supervisors of FMS, *International Journal of Advanced Manufacturing Technology*, 35, 150–168.

Uzam, M., and Zhou, M.C. (2006). An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems, *International Journal of Production Research*, 44(10), 1987–2030.

Yamalidou, K., Moody, J., Lemmon, M., and Antsaklis, P. (1996). Feedback control of Petri Nets based on place invariants, *Automatica*, 32(1), 15–28.

Zareiee, M., and Dideban, A. (2011). Reducing the number of constraints in non safe Petri Net, *In proc. International conference on modeling and simulation*, Dubai.

Zareiee, M., Dideban, A., and Nazemzadeh, P. (2011). From forbidden states to linear constraints, *In proc. International conference on modeling and simulation*.

Zareiee, M., Dideban, A., Orouji, A.A., and soltanizadeh, H. (2014). Solving the problem of forbidden states in Discrete Event Systems: A novel systematic method for reducing the number of control places, *Asian Journal of Control*.