A General Approach for the Computation of a Liveness Enforcing Supervisor for the Petri Net Model of an FMS

M. Uzam[1], Z. W. Li[2] and U.S. Abubakar[3]

[1]Meliksah Universitesi Mühendislik-Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümü, 38280, Talas, Kayseri, Turkey (Corresponding author. phone: +90-352-2077300, ext. 7351; fax: +90-352-2077337; e-mail: murat uzam@meliksah.edu.tr)
[2]Faculty of Information Technology, Macau University of Science and Technology, Taipa, Macau and aslo with the School of Electro-Mechanical Engineering, Xidian University, Xi'an, Shaanxi 710071, China (e-mail: zhwli@xidian.edu.cn)
[3]Meliksah Universitesi Mühendislik-Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümü, 38280, Talas, Kayseri, Turkey. (e-mail: ufs493@yahoo.co.uk)

Abstract. In this paper, a general approach is proposed for the computation of a liveness enforcing supervisor for the Petri net model of a flexible manufacturing system (FMS) prone to deadlocks. The proposed method is applicable to a lot of PN classes. A global sink/source place (GP) is used temporarily in the design steps and is finally removed when the liveness of the system is achieved. The aim is to obtain an easy to design deadlock prevention policy for PN models of FMSs that ensures liveness with optimal or near optimal permissiveness while maintaining the necessary computations simple.

Keywords: Discrete Event Systems, Petri Nets, Flexible Manufacturing Systems, Deadlock, Liveness Enforcing.

1 Introduction

Flexible manufacturing systems (FMS) are widely used by manufacturers. In an FMS, in order to finish pre-established operations, different processes compete for the limited number of shared resources such as buffers, fixtures, robots, automated guided vehicles (AGV), and other material-handling devices. Thus, this competition may result in deadlocks, a highly undesirable situation in which some processes keep waiting indefinitely for the other processes to release resources. Recently, a lot of work has been done to deal with the deadlock problem in FMSs [1-12].

Petri nets are widely used for the modeling of FMS due to their ability to easily detect the good behavior of a system like boundedness and deadlock-freeness [1]. A live Petri net guarantees deadlock-free operations. There are mainly two Petri net analysis techniques used to deal with deadlock prevention in FMS: structural analysis and reachability graph (RG) analysis. The examples of using the former may be found in [3, 5, 6] for different classes of FMS. The deadlock prevention control policy is obtained based on the characterization of the liveness in terms of Petri net items, i.e.,

siphons. The control policy can be implemented by adding control places (monitors) with initial marking and related arcs, to the initial Petri net model (PNM) of an FMS. In general the controlled model obtained in this case is suboptimal. The examples of using the latter may be found in [4, 7, 8, 10]. In this case, the RG of a PNM is used to obtain the live system behavior. The problem with these methods is that for very big PNs the computation of the monitors becomes very difficult to carry out due to the "state explosion problem". To tackle this problem, a first-met-bad-marking (FBM) based computationally efficient method is proposed in [10]. It was claimed in [9] that deadlock prevention based on the divide-and-conquer strategy is computationally superior compared with the well-established traditional global-conquer techniques such as [7, 8]. To develop a computationally efficient method of liveness-enforcing supervisors, the work in [11] presents a divide-and-conquer strategy for the computation of liveness enforcing supervisors (LES) from submodels for a large scale net system. Although the divide-and-conquer strategy proposed in [11] improves the traditional RG based methods proposed in [7, 8], it is necessary to deal with too many submodels when the number of shared resources is very big. Therefore the main purpose of this paper is to propose a general approach for the computation of a liveness enforcing supervisor for the Petri net model of an FMS without the necessity to divide a given PN model into its submodels. The proposed method is computationally efficient and provides optimal or near optimal permissive behavior for FMSs.

The rest of this paper is organized as follows. Some basic Petri net definitions used in this paper are briefly reviewed in Section 2. A general synthesis approach for liveness enforcement in Petri net models of FMS is proposed in section 3. An illustrative example is given in section 4, to show the applicability of the proposed method. Finally, some conclusions and directions for further research are provided in section 5.

2 Basics of Petri Nets

In this paper, Petri nets are used to model the flow of products in an FMS. Petri nets as a mathematical tool have a number of properties. When interpreted in the context of modeled manufacturing system, these properties allow one to identify the presence or absence of the functional properties of the system. In this section, some definitions and concepts which are related to this paper are briefly reviewed.

A Petri net is a five-tuple, $PN = (P, T, F, W, M_0)$ where: $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places, where $m > 0$; $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, where $n > 0$, with $P \cup T \neq \emptyset$ and $P \cap T \neq \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is the set of all directed arcs, where $P \times T \rightarrow N$ is the input function that defines the set of directed arcs from P to T, and $T \times P \rightarrow N$ is the output function that defines the set of directed arcs from T to P, where $N = \{0, 1, 2, \ldots\}$ is a set of non-negative integers, W: $F \rightarrow N$ is the weight function. M_0: $P \rightarrow N$ is the initial marking. The set of input (resp., output) transitions of a place p is denoted by ${}^\bullet p$ (resp., $p^\bullet$). Similarly, the set of input (resp., output) places of a transition t is denoted by ${}^\bullet t$ (resp., $t^\bullet$). A Petri net structure (P, T, F, W) without any specific initial marking is denoted by G. A Petri net with the given initial marking is denoted by (G, M_0). A transition t is said to be enabled or firable if

each input place $p \in {}^{\bullet}t$ is marked with at least $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t. A transition may fire if it is enabled. The firing of an enabled transition t removes $w(p,t)$ tokens from each input place $p \in {}^{\bullet}t$, and adds $w(t,p)$ tokens to each output place $p \in t^{\bullet}$, where $w(t,p)$ is the weight of the arc from t to p. This process is denoted by $M\,[t > M'$. The marking M of a Petri net indicates the number of tokens in each place, which represents the current state of the modelled system. When a marking M' is reached from a marking M by firing a sequence of transitions $\sigma = t_0t_1t_2...t_k$, this process is then denoted by $M\,[\sigma > M'$. The set of all reachable markings for a Petri net with initial marking M_0 is denoted by $RM(G,M_0)$.

A pair of a place p and a transition t is called a *self-loop* if both $p \in t^{\bullet}$ and $p \in {}^{\bullet}t$ hold. A Petri net is said to be *pure* if it has no self-loops. A Petri net is said to be *ordinary* if the weight of each arc is 1. A Petri net G is called *k-bounded*, or simply *bounded* if for every reachable marking $M \in RM(G,M_0)$, the number of tokens in any place p, $\forall p \in P$, is not greater than a finite number k, i.e. $M(p) \leq k$. A place p is called k-bounded, if the number of tokens in it is not greater than k. A Petri net G is called *safe*, if it is 1-bounded. A 1-bounded place p is called a safe place. Places are frequently used to represent buffers, tools, pallets, and AGVs in manufacturing systems. Boundedness is used to identify the existence of overflows in the modelled system. When a place models an operation, its safeness guarantees that the controller will not attempt to initiate an on-going process. A transition t is said to be *live* if for any $M \in RM(G,M_0)$, there exists a sequence of transitions firable from M which contains t. A Petri net G is said to be *live* if all the transitions are live. A Petri net G contains a *deadlock* if there is a marking $M \in RM(G,M_0)$ at which no transition is enabled. Such a marking is called a dead marking. Deadlock situations are a result of inappropriate resource allocation policies or exhaustive use of some or all resources. Liveness of a Petri net means that for each marking $M \in RM(G,M_0)$ reachable from M_0, it is finally possible to fire any transition t, $\forall t \in T$, in the Petri net through some firing sequence. This means that a live Petri net guarantees *deadlock-free* operations, no matter what firing sequence is chosen, i.e. if a Petri net is live, then it has no deadlock. A Petri net (G, M_0) is said to be *reversible*, if for each marking $M \in RM(G,M_0)$, M_0 is reachable from M. Thus, in a reversible net it is always possible to go back to initial marking (state) M_0. Many systems are required to return from the failure states to the preceding correct states. Thus the reversibility property is important to manufacturing system error recovery. This property also guarantees cyclic behavior for all repetitive manufacturing systems. Moreover, if a net contains a deadlock, then it is not reversible. A marking M' is said to be a *home state*, if for each marking $M \in RM(G,M_0)$, M' is reachable from M. Reversibility is a special case of the home state property, i.e. if the home state $M' = M_0$, then the net is reversible.

3 A General Approach for Liveness Enforcing in Petri Net Models of FMS

In this section, a general method is proposed for the computation of a liveness enforcing supervisor for a given Petri net model (PNM) of an FMS prone to deadlocks. There are usually three groups of the places in a PNM of an FMS: resource places (P_R),

activity places (P_A) and sink/source places ($P_{S/S}$). Resource places represent either shared or non-shared resources and initially there are tokens in these places representing the number of available resources. Activity places represent an action to process a part in a production sequence by a resource (machine, robot, etc.) and initially there are no tokens in these places. Initially, tokens deposited in sink/source places represent the maximal possible number of concurrent activities that can take place in a production sequence. In cyclic models a sink place is also a source place and vice versa.

The proposed method is applicable to a lot of Petri net classes currently available in the related literature. All computed monitors have ordinary arcs due to the proposed method. The algorithm provided below is used to compute the control places (monitors) based on the PNM. In the monitor computation steps of the proposed algorithm, a global sink/source place (GP) is used to make the necessary computation easily in an iterative way. The input transitions of the GP are input transitions of all sink/source places $P_{S/S}$. Likewise output transitions of the GP are output transitions of all sink/source places $P_{S/S}$. At each iteration, the reachability graph (RG) of the related net is computed. If the net is not live, then the RG is divided into a dead zone (DZ) and a live zone (LZ). The former may contain deadlock states (markings), and states which are inevitable lead to deadlocks or livelocks. The latter constitutes remaining good states of the RG representing the optimal system behavior. The control policy is based on the exclusion of the DZ from the RG, while making sure that every state within the LZ can still be reached. All states in the DZ are considered as bad markings (BM). From a BM, only the markings of activity places are considered. Then, the objective is to prevent the marking of the subset of the activity places of the BM from being reached. Therefore, the marking of the subset of the activity places is characterized as a place invariant (PI) of the PNM. In the PI relating to a BM, the sum of tokens within the subset of the activity places has to be at most one token less than their current value within the BM in order not to reach the BM. A PI can be implemented by a control place with its related arcs and initial marking. Here the simplified version [7] of the method proposed in [13] is used in order to obtain a control place (monitor) from a PI. The redundancy test of [12] is adopted to find out if there are any redundant monitors within computed control places in the computation procedures. Finally, a live controlled Petri net is obtained by including all necessary control places in the PNM. Although not explained in the algorithm, in order to simplify big PNMs so as to make necessary computation easily as in [2], the Petri net reduction approach may be used. The reachability graph analysis of PNMs can be carried out by currently available Petri net analysis tools. In this work, INA [14] is used. For a given PNM suffering from deadlocks (and/or livelocks) INA provides both LZ and DZ. The former is the first strongly connected component, and the latter is the strongly connected components other than the first one. In this work, the DZ is then considered as the collection of all bad markings (BM_i, i= 1, 2, . . .).

Algorithm: Synthesis of Liveness Enforcing Supervisor for Petri Net Models

Input: A Petri net model (PNM) of an FMS prone to deadlocks
Output: Liveness enforcing supervisor for the PNM

Step 1. Define input and output transitions of all sink/source places $P_{S/S}$. Add a global sink/source place (GP) to the PNM. Thus a new net system denoted as N_B = PNM + GP is obtained, where $B \in N = \{1, 2, \ldots\}$.

Step 2. for ($B = 1$; $B \leq K$; B ++)
/* B is the number of tokens in GP and K is the sum of initial tokens in all sink/source places $P_{S/S}$ */
{

2.B.1. Compute the reachability graph RG_B of N_B.
If N_B is live,
Then consider net N_B with B:=B+1, i.e., go to step 2.B.1).
Else define the LZ_B and DZ_B of RG_B.

2.B.2. Define a PI for each bad marking (BM) within DZ_B, from the subset of marked activity places of BM.

2.B.3. Compute a monitor C for each PI using the simplified invariant-based control method.

2.B.4. If the number of monitors computed for N_B is greater than 1, then carry out the redundancy test by using the method proposed in [5] to find out the set of necessary monitors $C_{B,i}$; $i = 1, 2, 3, \ldots$

2.B.5. Augment all necessary monitors computed in the previous step into N_B (N_B: = $N_B + C_{B,i}$, $i = 1, 2, 3, \ldots$).
}

Step 3. Obtain a live controlled PNM by augmenting all necessary monitors computed in step 2 into the PNM.

Step 4. Exit

end of Algorithm.

4 Illustrative Example

In order to show the applicability of the proposed synthesis approach, in this section an example is considered. Fig. 1 shows an L-S^3PR net taken from [13]. In this PNM there are eight activity places P_A = {p2–p5, p7–p10}, four shared resource places P_R = {p11–p14}, and two sink/source (idle) places $P_{S/S}$ = {p1, p6}. The RG of this PNM contains 119 states, 27 of which are bad states. Then, the optimal solution should provide a live system behavior with 92 good states. Let us now apply the proposed method to this PNM.

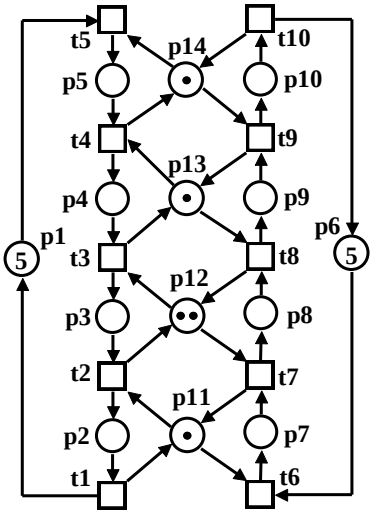

Figure 1. An L-S^{3}PR net taken from [16].

Step 1. Input transitions of sink/source places p1 and p6 are $^\bullet$p1= {t1} and $^\bullet$p6 = {t10}. Likewise output transitions of p1 and p6 are p1$^\bullet$ = {t5} and p6$^\bullet$ = {t6}. Therefore input and output transitions of the GP are $^\bullet$GP = {t1, t10} and GP$^\bullet$ = {t5, t6}. When the GP is added within the PNM, a new net structure N_B = PNM + GP is obtained as shown in Fig. 2.

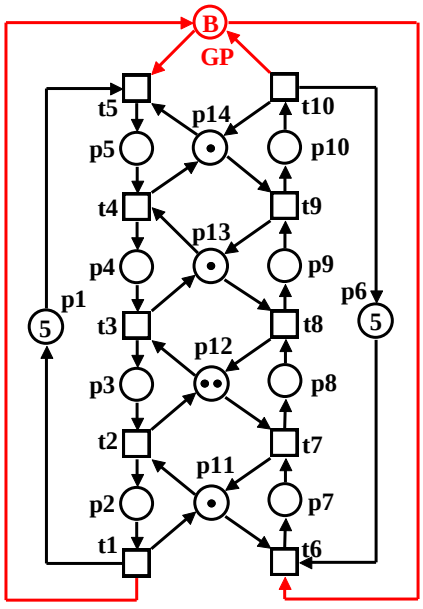

Figure 2. The net N_B; N_B = PNM + GP.

Step2.

Step 2.1.1. (B = 1). When one token is deposited in the GP, the net N_1 shown in Fig. 3 is obtained.

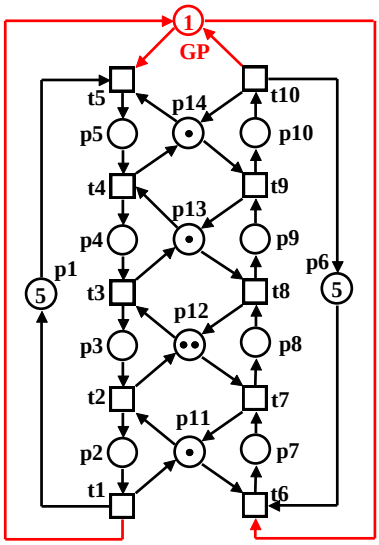

Figure 3. The net N_1 (B = 1).

The net N_1 is live with 7 good states. B:= B+1 = 2.

Step 2.2.1. (B = 2). When two tokens are deposited in the GP, the net N_2 shown in Fig. 4, is obtained.

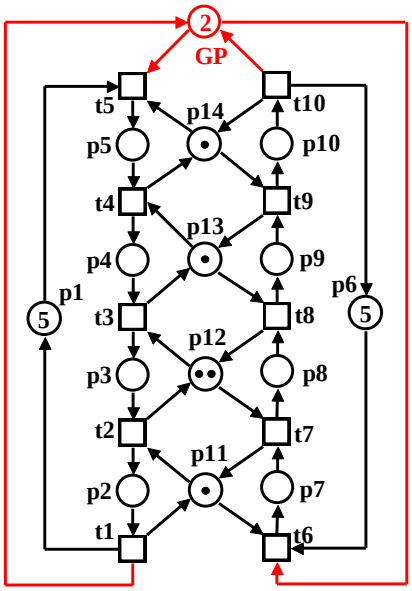

Figure 4. The net N_2 (B = 2).

The net N_2 is not live. There are 35 states within the RG_2 of N_2. DZ_2 includes 1 bad state, i.e., BM_1 and LZ_2 contains 34 good states.

Step 2.2.2. The markings of the activity places of BM_1 are shown in Table 1.

Table 1. The markings of activity places of BM_1.

State number	p2	p3	p4	p5	p7	p8	p9	p10
s_{18}	0	0	0	1	0	0	1	0

In order not to reach BM_1 the place invariant $PI_1 = \mu_5 + \mu_9 \leq 1$ is established.

Step 2.2.3. In order to enforce PI_1, monitor C_1 is computed as shown in Table 2.

Table 2. Computed monitor C_1 for PI_1.

C_i	${}^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
C_1	t4, t9	t5, t8	1

Step 2.2.4. There is no need for redundancy test as there is only one computed monitor.

Step 2.2.5. When C_1 is augmented in the uncontrolled model N_2, the controlled N_2 is obtained as follows: $N_2 := N_2 + C_1$ and is shown in Fig. 5.

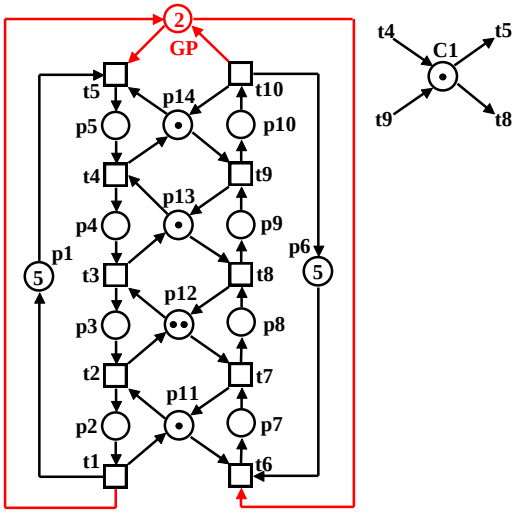

Figure 5. The controlled N_2 ($N_2 := N_2 + C_1$).

It is verified that the controlled model N_2 shown in Fig. 5 is live with 34 good states. This is the optimal live behavior for the controlled N_2.

Step 2.3.1. (B:= B+1 = 3). The net N_3, shown in Fig. 6, is obtained by increasing the number of tokens in the GP within the controlled N_2.

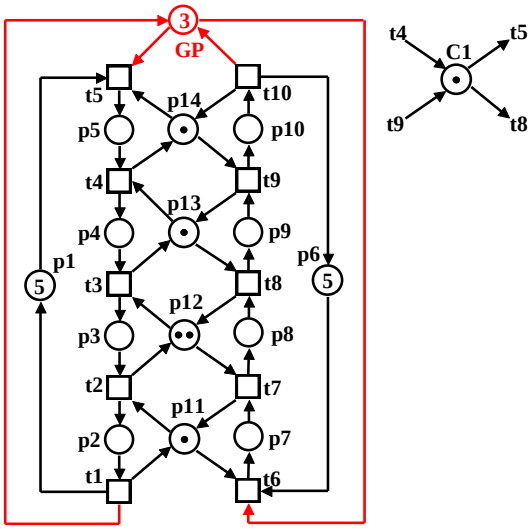

Figure 6. The net N_3.

The net N_3 is not live. There are 73 states within the RG_3 of N_3. DZ_3 includes 3 bad states BM_1, BM_2 and BM_3, and LZ_3 contains 70 good states.

Step 2.3.2. The markings of the activity places of BM_2, BM_3 and BM_4 are shown in Table 3.

Table 3. The markings of activity places of BM_2, BM_3 and BM_4.

State number	p2	p3	p4	p5	p7	p8	p9	p10
s_{17}	0	2	0	0	1	0	0	0
s_{43}	0	0	1	0	0	2	0	0
s_{44}	0	0	0	1	0	2	0	0

In order not to reach BM_2, BM_3 and BM_4 the following place invariants are established respectively: $PI_2 = \mu_3 + \mu_7 \leq 2$, $PI_3 = \mu_4 + \mu_8 \leq 2$, $PI_4 = \mu_5 + \mu_8 \leq 2$.

Step 2.3.3. Monitors C_2, C_3 and C_4 are computed in order to enforce PI_2, PI_3 and PI_4 as shown in Table 4.

Table 4. Computed monitors C_2, C_3 and C_4.

C_i	${}^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
C_2	t2, t7	t3, t6	2
C_3	t3, t8	t4, t7	2
C_4	t4, t8	t5, t7	2

Step 2.3.4. The redundancy test shows that computed monitors C_2, C_3 and C_4 are all necessary.

Step 2.3.5. When C_2, C_3 and C_4 are augmented in the uncontrolled model N_3, the controlled N_3 is obtained as follows: $N_3 := N_3 + C_2 + C_3 + C_4$ and is shown in Fig. 7.

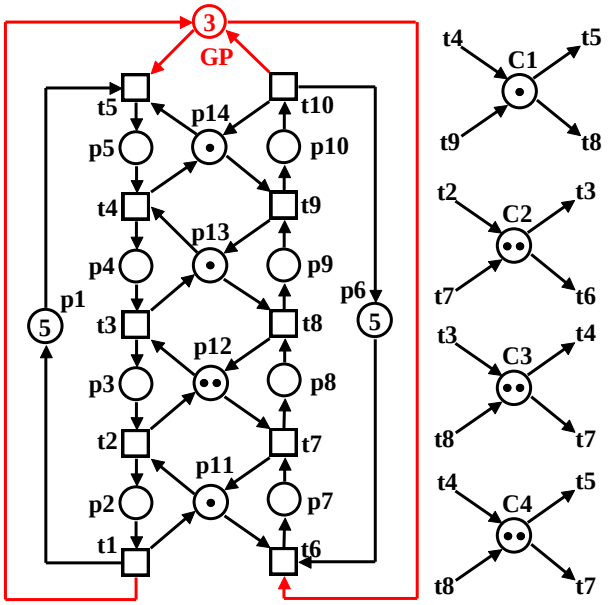

Figure 7. The controlled N_3 ($N_3 := N_3 + C_2 + C_3 + C_4$).

It is verified that the controlled model N_3 shown in Fig. 7 is live with 70 good states. This is the optimal live behavior for the controlled N_3.

Step 2.4.1. (B:= B+1 = 4). The net N_4, shown in Fig. 8, is obtained by increasing the number of tokens in the GP within the controlled N_3.

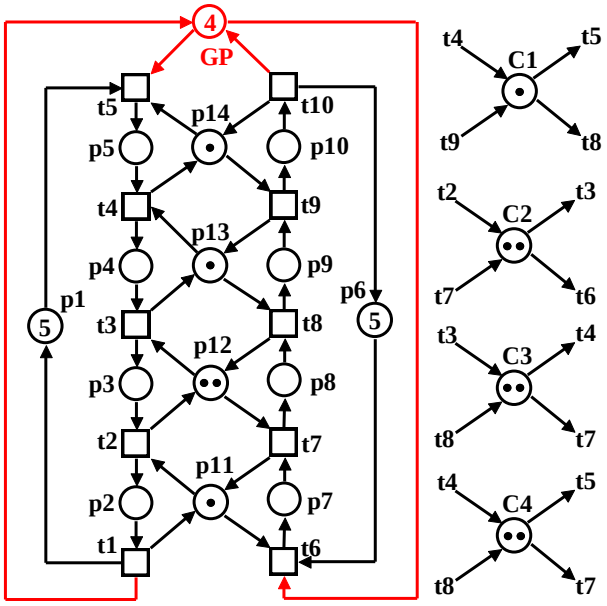

Figure 8. The net N_4.

The net N_4 is not live. There are 91 states within the RG_4 of N_4. DZ_4 includes 3 bad states BM_5, BM_6 and BM_7 and LZ_4 contains 88 good states.

Step 2.4.2. The markings of the activity places of BM_5, BM_6 and BM_7 are shown in Table 5.

Table 5. The markings of activity places of BM_5, BM_6 and BM_7.

State number	p2	p3	p4	p5	p7	p8	p9	p10
s_{41}	0	1	0	1	1	1	0	0
s_{42}	0	1	1	0	1	1	0	0
s_{83}	0	0	1	1	1	1	0	0

In order not to reach BM_5, BM_6 and BM_7 the following place invariants are established respectively: $PI_5 = \mu_3 + \mu_5 + \mu_7 + \mu_8 \leq 3$, $PI_6 = \mu_3 + \mu_4 + \mu_7 + \mu_8 \leq 3$, $PI_7 = \mu_4 + \mu_5 + \mu_7 + \mu_8 \leq 3$.

Step 2.4.3. Monitors C_5, C_6 and C_7 are computed in order to enforce PI_5, PI_6 and PI_7 as shown in Table 6.

Table 6. Computed monitors C_5, C_6 and C_7.

C_i	${}^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
C_5	t2, t4, t8	t3, t5, t6	3
C_6	t2, t8	t4, t6	3
C_7	t3, t8	t5, t6	3

Step 2.4.4. The redundancy test shows that computed monitors C_5, C_6 and C_7 are all necessary.

Step 2.4.5. When C_5, C_6 and C_7 are augmented in the uncontrolled model N_4, the controlled N_4 is obtained as follows: $N_4 := N_4 + C_5 + C_6 + C_7$ and is shown in Fig. 9.

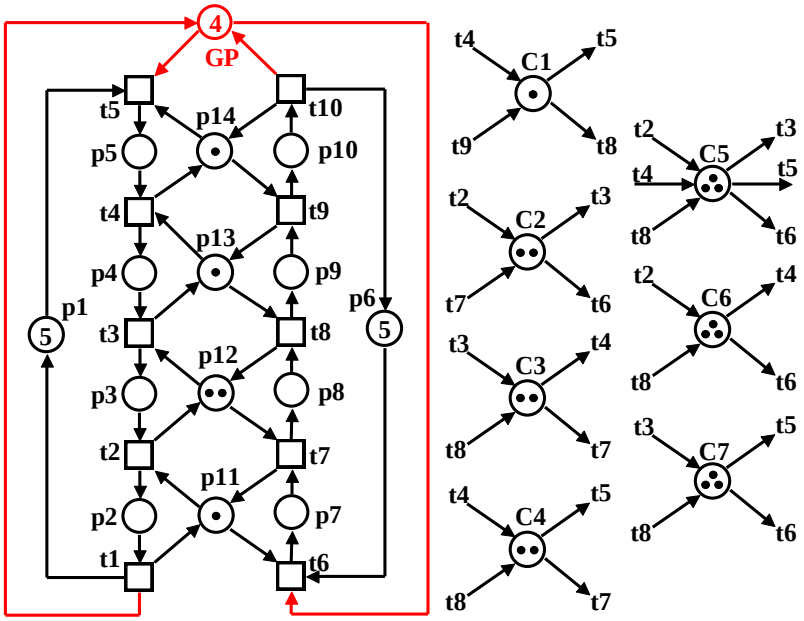

Figure 9. The controlled N_4 (N_4:= $N_4 + C_5 + C_6 + C_7$).

It is verified that the controlled model N_4 shown in Fig. 9 is live with 88 good states.

Step 2.5.1. (B:= B+1 = 5). The net N_5, shown in Fig. 10, is obtained by increasing the number of tokens in the GP within the controlled N_4.

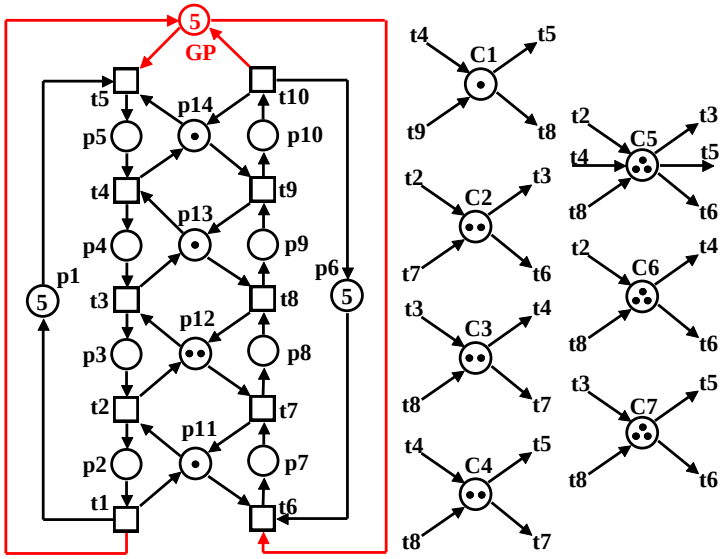

Figure 10. The net N_5 (B = 5).

The net N_5 is live with 92 good states. This is the optimal solution for both the N_5 and the original uncontrolled PNM.

Step 3. The live controlled PNM shown in Fig. 11 is obtained by augmenting seven monitors, computed in step 2 and provided in Table 7, into the uncontrolled model PNM shown in Fig. 1. It is live with 92 good states. The liveness enforcing procedure applied for the PNM is provided in Table 8. In this table the last column shows the number of unreachable states. As the elements of this column are all zero this indicates that there are no good states lost due to included monitors.

Table 7. Monitors computed for the L-S^3PR net.

C_i	${}^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
C_1	t4, t9	t5, t8	1
C_2	t2, t7	t3, t6	2
C_3	t3, t8	t4, t7	2
C_4	t4, t8	t5, t7	2
C_5	t2, t4, t8	t3, t5, t6	3
C_6	t2, t8	t4, t6	3
C_7	t3, t8	t5, t6	3

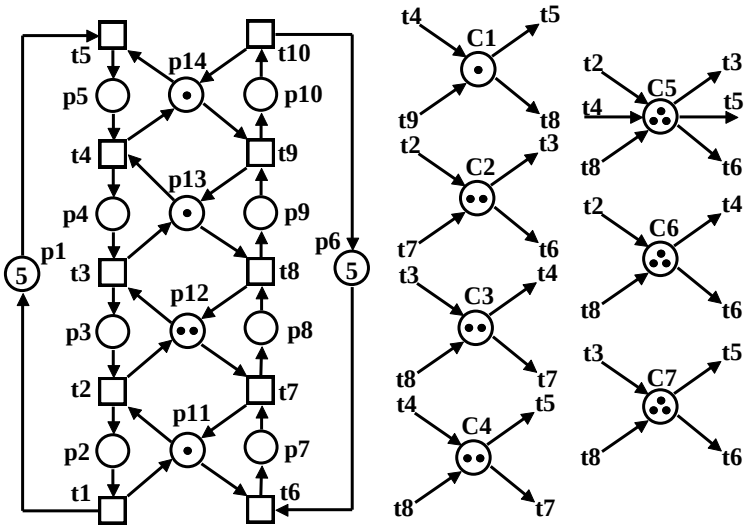

Figure 11. Optimally controlled live PNM.

Table 8. Liveness enforcing procedure applied to the *L-S^{3}PR* net.

B	included C	Is the net live?	# states in		
			RG	*DZ*	*LZ*
1	-	Yes	9	-	9
2	-	No	35	1	34
3	*C1*	No	73	3	70
4	*C1, ..., C4*	No	91	3	88
5	*C1, ..., C7*	Yes	92	0	92

Computed monitor C	# states in the controlled PNM RG=LZ	UR
-		
C1	34	*0*
C2, C3, C4	70	*0*
C5, C6, C7	88	*0*
-		

5 Conclusions

In this paper, a general method is proposed to obtain an optimal or a near optimal solution for the synthesis of liveness enforcing supervisors in flexible manufacturing systems modeled with Petri nets. The applicability of the proposed approach is shown by means of an example from the literature. The method is easy to use and provides very high behavioral permissiveness. The proposed method is applicable as is to a lot of Petri net classes currently available in the literature without modifications. Therefore further publications will be provided to show the applicability of the proposed method to different classes of Petri nets. Further research will also be conducted to improve the behavioral permissiveness of the proposed approach for generalized Petri net models such as S^4R or S^4PR.

Acknowledgements

This work was supported by the research grant of the Scientific and Technological Research Council of Turkey (Türkiye Bilimsel ve Teknolojik Araştırma Kurumu) under the project number TÜBİTAK-112M229, and the Science and Technology Development Fund, MSAR, under Grant No. 066/2012/A2, and National Natural Science Foundation of China under Grant No. 61374068

References

1. Z.W. Li, N. Wu, and M.C. Zhou., "Deadlock control of automated manufacturing systems based on Petri nets_a literature review," IEEE Transactions on System, Man, and Cybernetics, Part C; Applications and Reviews, pp. 432-462, July 2012.
2. Z.W. Li, M.C. Zhou, and N.Q. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," IEEE Trans. on Sys. Man and Cybern. Part C-Applications and Reviews vol. 38, Issue: 2, pp. 173-188, March 2008.
3. J. Ezpeleta, J.M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," IEEE Trans. Robot. Automat., vol. 11, Issue: 2, pp. 173-184, 1995.
4. M. Uzam, "An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions," Int. J. Adv. Manuf. Tech., vol. 19, Issue: 3, pp. 192-208, 2002.
5. Z.W. Li and M.C. Zhou, "Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems," IEEE Trans. Robot. Sys. Man and Cyber. Part A: Systems and Humans, vol. 34, Issue: 1, pp. 38- 51, 2004.
6. Y.S. Huang, M.D. Jeng, X.L. Xie, et al., "Siphon-based deadlock prevention policy for flexible manufacturing systems," IEEE Trans. on Sys. Man and Cybern. Part A-Systems and Humans vol. 36, Issue: 6, pp. 1248-1256, November 2006.
7. M. Uzam and M.C. Zhou, "An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems," Int. J. Prod. Res., vol. 44, Issue: 10, pp. 1987-2030, 15 May 2006.
8. M. Uzam and M.C. Zhou, "An iterative synthesis approach to Petri net based deadlock prevention policy for flexible manufacturing systems," IEEE Trans. on Sys., Man and Cybern. - Part A: Systems and Humans, vol. 37, Issue: 3, pp. 362-371, 2007.
9. Z.W. Li, S. Zhu, and M.C. Zhou, "A Divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems," IEEE Trans. on Sys. Man and Cybern. Part C-Applications and Reviews vol. 39, Issue: 2, pp. 156-169, March 2009.
10. Y.F. Chen, Z.W. Li, M. Khalgui, O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," IEEE Trans. on Auto. Sci. and Eng. vol. 8, Issue: 2, pp. 374-39,3 April 2011.
11. M. Uzam, R. S. Zakariyya, Z. W. Li and G. Gelen, "The computation of liveness enforcing supervisors from submodels of a Petri net model of FMSs," IEEE TENCON 2013, October 22-25, Xi'an, China, pp. 1–4, 2013.
12. M. Uzam, Z.W. Li, and M.C. Zhou, "Identification and elimination of redundant control places in Petri net based liveness enforcing supervisors of FMS," The Int. J. of Adv. Manuf. Tech., vol. 35, Issues: 1-2, pp. 150-168, 2007.
13. K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of petri nets based on place invariants," Automatica, vol. 32, Issue:1, pp. 15-28, 1996.
14. INA, 31.07.2003, Integrated Net Analyzer, a software tool for analysis of Petri nets, Version 2.2. Posted at URL: http://www.informatik.hu-berlin.de/~starke/ina.html.