Article

Single Controller-Based Colored Petri Nets for Deadlock Control in Automated Manufacturing Systems

Husam Kaid [1],*, Abdulrahman Al-Ahmari [1],*, Zhiwu Li [2,3] and Reggie Davidrajuh [4]

[1] Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[2] Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China; systemscontrol@gmail.com
[3] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[4] Faculty of Science and Technology, University of Stavanger, 4036 Stavanger, Norway; reggie.davidrajuh@uis.no
* Correspondence: yemenhussam@yahoo.com (H.K.); alahmari@ksu.edu.sa (A.A.-A.)

check for updates

Abstract: Deadlock control approaches based on Petri nets are usually implemented by adding control places and related arcs to the Petri net model of a system. The main disadvantage of the existing policies is that many control places and associated arcs are added to the initially constructed Petri net model, which significantly increases the complexity of the supervisor of the Petri net model. The objective of this study is to develop a two-step robust deadlock control approach. In the first step, we use a method of deadlock prevention based on strict minimal siphons (SMSs) to create a controlled Petri net model. In the second step, all control places obtained in the first step are merged into a single control place based on the colored Petri net to mark all SMSs. Finally, we compare the proposed method with the existing methods from the literature.

Keywords: Automated manufacturing system; colored Petri net; deadlock prevention; siphon

1. Introduction

An automated manufacturing system (AMS) is a conglomeration of robots, machine tools, fixtures, and buffers. Several types of products enter the manufacturing system at separate points in time; the system can process these parts based on a specified sequence of operations and resource sharing. The sharing of resources leads to the occurrence of deadlock states, in which the local or global system is incapacitated [1–4]. Thus, an efficient deadlock-control algorithm is needed to prevent the deadlocks in an AMS. Petri nets are an excellent mathematical and graphical tool suitable for modeling, analyzing, and controlling deadlocks in AMSs [5,6]. The behavior and characteristics of an AMS (such as synchronization, conflict, and sequences) can be described by using Petri nets. Moreover, Petri nets may be used to provide the liveness and boundedness of a system [7]. To address the deadlock problem in AMSs, several approaches with Petri nets exist. These approaches are categorized into three strategies: (1) deadlock detection and recovery, (2) deadlock prevention, and (3) deadlock avoidance [7,8].

Traditionally, deadlock control approaches for AMS control are evaluated by using three criteria: structural complexity, computational complexity, and behavioral permissiveness [7]. Structural complexity means that a controller can be implemented with fewer monitors and arcs. When the computational complexity of a deadlock control approach is low, it can be applied to a large-scale system [7]. Behavioral permissiveness achieves high resource utilization in a controlled Petri net.

Deadlock control may be implemented in AMS with reliable resources (resources without failures or breakdowns) or unreliable resources (resources with failures or breakdowns). For reliable resources, there are two main techniques to prevent deadlocks in AMSs using a Petri net: reachability graph analysis [9–11] and structural analysis [12,13]. The reachability graph analysis needs listing all or a part of the reachable markings of the Petri net model. There are two parts of the reachability graph: the deadlock zone (DZ) and the live zone (LZ). First-met bad markings (FBMs) are defined in and extracted from the DZ. In this case, the deadlock is eliminated by designing and adding a monitor to prohibit the first-met bad markings from being reached. In this process, all first-met bad markings can be prevented by using iterations [14]. Several policies have been developed to prevent deadlock states, including iterative methods, the theory of region, and siphon control [10,13–19]. The weakness of the reachability graph analysis is that the size of a reachability graph of a Petri net grows quickly and, in the worst case, grows exponentially with respect to the net size and its initial marking, and the net can easily reach an unmanageable level. Structural analysis is often applied to a typical structure of Petri nets, such as siphons. The control steps in this technique are simple: each minimal siphon is prohibited from being non-empty, and each unmarked minimal siphon needs an added monitor to ensure a system to be live. However, the weakness of this technique is that the number of control places will be increased when the size of a Petri net model is increased; hence, this results in high structural complexity [20].

In the literature, deadlock control approaches based on the structural analysis technique (siphons) for AMSs with the Petri nets framework can be implemented by inserting the control places and the associated arcs to the original net, so that its siphons are permanently non-empty. The main disadvantage of the current policies is that many control places and associated arcs are inserted into the original Petri net model, which leads to the increased complexity of the supervisor of the Petri net model, compared with the initial model for the Petri net supervisor. Hence, an efficient approach is needed to minimize the Petri net supervisors' structural complexity for AMS. The objective of this study is to develop a two-step robust deadlock control policy. A technique based on SMSs developed in [21] is used in the first phase to develop a controlled Petri net model. In the second step, all control places obtained in the first step are merged into one control place based on colored Petri nets to make all SMSs marked.

The rest of the paper is organized as follows. Basic concepts of Petri nets are introduced in Section 2. Section 3 describes a deadlock prevention approach based on the SMS and the proposed robust control based on colored Petri nets. Section 4 shows an example from the literature. Finally, Section 5 presents conclusions and future research.

2. Preliminaries

This section introduces the basics of Petri nets and a general Petri net simulator (GPenSIM) tool.

2.1. Basics of Petri Nets

Let $N = (P, T, F, W)$ be a Petri net, where P and T are finite non-empty sets of places and transitions, respectively. Elements in $P \cup T$ are named nodes. Here, $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$; P and T are depicted by circles and bars, respectively. Next, $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs that join the transitions with places (and vice versa), $W: (P \times T) \cup (T \times P) \to IN$ is a mapping that assigns an arc's weight, where $IN = \{0, 1, 2, \dots \}$.

N is known as an ordinary net if $\forall (p, t) \in F$, $W(p, t) = 1$, where $N = (P, T, F)$. N is named a weighted net if there is an arc between x and y such that $W(x, y) > 1$. Let $N = (P, T, F, W)$ and node $a \in P \cup T$. Then, $\cdot a = \{b \in P \cup T \mid (b, a) \in F\}$ is named the preset of node a, and $a \cdot = \{b \in P \cup T \mid (a, b) \in F\}$ is named the postset of node a.

A marking M of N is a mapping $M: P \to IN$. Next, (N, M_o) is a marked Petri net (PN), represented as $PN = (P, T, F, W, M_o)$, where the initial marking of PN is $M_o: P \to IN$. A transition $t \in T$ is enabled at marking M if for all $p \in \cdot t$, $M(p) \geq W(p, t)$, which is denoted as $M[t\rangle$. When a transition t fires, it takes

$W(p, t)$ token (s) from each place $p \in \cdot t$, and adds $W(t, p)$ token (s) in each place $p \in t\cdot$. Thus, it reaches a new marking M', denoted as $M[t\rangle M'$, where $M'(p) = M(p) - W(p, t) + W(t, p)$.

We call N self-loop free if for all $a, b \in P \cup T$, $W(a, b) > 0$ implies $W(b, a) = 0$. Let $[N]$ be an incidence matrix of net N, where $[N]$ is an integer matrix that consists of $|T|$ columns and $|P|$ rows with $[N](p, t) = W(t, p) - W(p, t)$. The set of markings that are reachable from M in N is named the set of reachability of the Petri net model (N, M) denoted by $R(N, M)$.

Let (N, M_o) be a marked Petri net. A transition $t \in T$ is live if for all $M \in R(N, M)$, there exists a reachable marking $M' \in R(N, M)$ such that $M'[t\rangle$ holds. A transition is dead at M_o if there does not exist $t \in T$ such that $M_o[t\rangle$ holds. M' is said to be reachable from M if there exists a finite transition sequence $\delta = t_1\,t_2\,t_3\,\dots\,t_n$ that can be fired, and markings $M_1, M_2, M_3, \dots$, and M_{n-1} such that $M[t_1\rangle M_1[t_2\rangle M_2[t_3\rangle M_2\dots M_{n-1}[t_n\rangle M'$, denoted as $M[\delta\rangle M'$, satisfies the state equation $M' = M + [N]\,\vec{\delta}$, where $\vec{\delta}: T \to \mathbf{IN}$ maps t in T to the number of appearances of t in δ and is called a Parikh vector or a firing count vector.

Let (N, M_o) be a marked Petri net. It is said to be k-bounded if for all $M \in R(N, M_0)$, for all $p \in P$, $M(p) \le k$ ($k \in \{1, 2, 3, \dots\}$). A net is safe if all of its places are safe, i.e., in each place p, the number of tokens does not exceed one. In other words, a net is k-safe if it is k-bounded.

P-vectors (place vectors) and T-vectors (transition vectors) are column vectors. A P-vector $I: P \to Z$ cataloged by P is said to be a place invariant or P-invariant if $I \neq \mathbf{0}$ and $I^T . [N] = \mathbf{0}^T$, and a T-vector $J: T \to Z$ cataloged by T is said to be a transition invariant or T-invariant if $J \neq \mathbf{0}$ and $[N] . J = \mathbf{0}$, where Z is the set of integers.

When each element of I is nonnegative, place invariant I is called a place semiflow or P-semiflow. Assume that I is a P-invariant of a net with (N, M_o) and M is a marking reachable from the initial marking M_o. Then, $I^T M = I^T M_o$. Let $\|I\| = \{p\,|I(p) \neq 0\}$ be the support of P-invariant I.

The supports of P-invariant I are classified into three types: (1) $\|I\|^+$ is the positive support of P-invariant I with $\|I\|^+= \{p|I(p) > 0\}$. (2) $\|I\|^-$ is the negative support of P-invariant I with $\|I\|^- = \{p\,|I(p) < 0\}$. (3) I is a minimal P-invariant if $\|I\|$ is not a superset of the support of any other one and its components are mutually prime. Let l_i be the coefficients of P-invariant I if for all $p_i \in P$, $l_i = I(p_i)$.

A colored Petri net (CPN) is described as a nine-tuple $CPN = (P, T, F, SC, C_f, N_f, A_f, G_f, I_f)$, where

1.　P and T are finite nonempty sets of places and transitions, respectively, assuming $P \cap T = \emptyset$. F is a set of flows (arcs), from $p_i \in P$ to $t_j \in T$ and from $t_i \in T$ to $p_j \in P$.
2.　SC is a color set that contains colors c_i and the operations on the colors.
3.　C_f is the color function that maps $p_i \in P$ into colors $c_i \in SC$.
4.　N_f is the node function that maps F into $(P \times T) \cup (T \times P)$.
5.　A_f is the arc function that maps each flow (arc) $f \in F$ into the term e.
6.　G_f is the guard function that maps each transition $t_i \in T$ to a guard expression g that has a Boolean value.
7.　I_f is the initialization function that maps each place $p_i \in P$ into an initialization expression.

2.2. GPenSIM Tool

GPenSIM was developed by R. Davidrajuh (the fourth author of our paper) and runs in MATLAB. GPenSIM has been designed to model, control, simulate, and analyze discrete event systems [22]. GPenSIM enables the integration of Petri net models with other toolboxes in MATLAB (e.g., "Control systems" and "Fuzzy logic"). Table 1 shows the advantages and disadvantages of GPenSIM compared to CPN Tools [23]. Compared to the CPN tools, it is simpler to create a colored Petri net in GPenSIM. For instance,

1.　Being versatile, CPN allows manipulation of the functions C_f, N_f, A_f, G_f, and I_f independently. However, being simple and crude, these functions (C_f, N_f, A_f, G_f, and I_f) are merged together

in GPenSIM and are coded in the preprocessor files. Hence, GPenSIM allows fewer degrees of freedom when developing a model.

2. In CPN tools, it is possible to impose logical constraints on places, transitions, and arcs. In GPenSIM, logical expressions can only be processed by transitions. Inevitably, this means GPenSIM poses restrictions in modeling compared to CPN. However, this is the price paid for achieving simplicity in GPenSIM (easiness in learning, using, and extending).

3. The arc weights can dynamically alter in CPN tools because of the logic conditions connected to it. GPenSIM does not allow dynamic nets (e.g., dynamic arcs, run-time removal of places and transitions). Once a Petri net is defined in the Petri net definition file (PDF), it cannot be changed.

Table 1. The advantages and disadvantages of GPenSIM compared to CPN Tools.

Tool	Advantages	Disadvantages
GPenSIM	1. Simple, easy to learn, and use. 2. Easy to extend. 3. GPenSIM runs on MATLAB, it is easy to interconnect with other toolboxes.	1. Limited functionality. 2. The user is supposed to extend the primitive functions offered or to develop their own functions.
CPN Tools	1. A large number of functions available. 2. Has been used to model large systems.	1. Quite complicated, as this is a product of several researchers, extending the tool into diverse directions over a period of 20 or more years. 2. Lack of user manual deprives new users.

To model, simulate, analyze, and control the Petri net models in GPenSIM, three files should be coded: Petri net definition file (PDF), main simulation file (MSF), and pre- and postprocessor files.

1. PDF defines the static elements of a Petri net (places, transitions, and arcs).

2. Before the simulation starts, MSF loads a PDF into memory and the workbench, and then the simulation begins. During the simulation runs, MSF will be blocked; the control will be handed back to MSF together with the simulation results when the simulation is finished. Consequently, MSF has no control over what happens during the simulation.

3. Pre-and postprocessors will be called during the simulation before and after firing of the transition. The preprocessor will inspect if the conditions of firing for a certain transition are met, and the postprocessor will execute post-firing activities if needed after a certain transition has been fired. These can be used to control the runtime of the system, as they are called during the simulation.

All tokens are homogeneous inside a place. It does not matter which token is first or last to arrive at the place. Similarly, it does not matter by which transition a token is deposited at the place. However, GPenSIM introduces the token colors. Each token can become identifiable and unique with a unique token identification number (**tokID**). Moreover, we can add some tags ("colors") to each token. The following problems are crucial when using colors in GPenSIM:

1. Only transitions can manipulate colors: the colors of the output tokens can be added, deleted, or changed in the preprocessor.

2. By default, colors are inherited: the system gathers all colored tokens from the input places when a transition fires and then transfers the colored tokens to the output places. However, color inheritance can be avoided by overriding.

3. An enabled transition may choose certain color-based input tokens.
4. An enabled transition may choose certain time-based input tokens (e.g., when the creation time of the tokens is known).
5. A token has the following structure: **tokID**, time of creation, and color setting.

     A. **tokID**: a single token identifier (integer value).
     B. **creation_time**: the transition time (real value) when the token was produced. Importantly, this time may differ when the transition actually deposited the token in an output place.
     C. **t_color** (text string set) is a color setting.

There are several GPenSIM functions used for color manipulation. One of the functions used in this study is **tokenEXColor**, which can be expressed as follows:

[set_of_tokID,nr_token_av] = tokenEXColor (place, nr_tokens_wanted, t_color), where the function requires three input arguments and returns two output values.

1. Input arguments:

(place, nr_tokens_wanted, t_color):

- **Place:** from which place the tokens are to be selected.
- **nr_tokens_wanted:** the number of required tokens with the specified color.
- **t_color:** a set of colors.

2. Output values:

[set_of_tokID,nr_token_av]

- **set_of_tokID:** a set of tokIDs that meet the color specifications. The set length of tokIDs is equal to the input argument of **nr_tokens_wanted**.
- **nr_token_av:** the number of valid tokIDs available in **set_of_tokID**; the set may have trailing zeros to match the length of **nr_tokens_wanted**.

3. Deadlock Prevention Policy Based on SMSs and Colored Petri Nets

In this section, we use a deadlock-prevention approach based on strict SMSs to design a controlled Petri net model. This approach is adopted from Ezpeleta et al. [1].

Definition 1 [23]. *A PN N = ($P_A \cup \{p^0\}$, T, F) is said to be a simple sequential process (S^2P), if: (1) N is a strongly connected state machine and (2) each circuit N includes place p^0, where p^0 is named the idle process place and $P_A \neq \emptyset$ is an operation places set.*

Definition 2 [23]. *A PN N = ($\{p^0\} \cup P_A \cup P_R$, T, F) is said to be a simple sequential process with resources (S^2PR) such that:*

1. *The subnet generated by X = $P_A \cup \{p^0\} \cup T$ is an S^2P.*
2. *$P_R \neq \emptyset$ and ($P_A \cup \{p^0\}$) $\cap P_R = \emptyset$, where P_R is a resource places set.*
3. *$\forall p \in P_A, \forall t \in \cdot p, \forall t' \in p\cdot, \exists r_p \in P_R, \cdot t \cap P_R = t'\cdot \cap P_R = \{r_p\}$.*
4. *$\forall r \in P_R, \cdot r \cap P_A = r\cdot \cap P_A \neq \emptyset$ and $\cdot r \cap r\cdot \neq \emptyset$.*
5. *$(p^0) \cap P_R = (p^0)\cdot \cap P_R \neq \emptyset$.*

Definition 3 [23]. *Let N = ($\{p^0\} \cup P_A \cup P_R$, T, F) be an S^2PR, and M_o be an initial marking of N. An S^2PR with such a marking is said to be acceptably marked if (1) $M_o(p^0) \geq 1$, $M_o(r) \geq 1$, $\forall r \in P_R$, and (3) $M_o(p) = 0$, $\forall p \in P_A$.*

Definition 4 [23]. *A system of S^2PR, named S^3PR for abbreviation, is defined recursively as follows:*

1.　*An S^2PR is as well an S^3PR*

2.　*Let N_1 and N_2 be two S^3PRs, where $N_1 = (\{p^0{}_1\} \cup P_{A1} \cup P_{R1}, T_1, F_1)$ and $N_2 = (\{p^0{}_2\} \cup P_{A2} \cup P_{R2}, T_2, F_2)$, such that $(\{p^0{}_1\} \cup P_{A1}) \cap (\{p^0{}_2\} \cup P_{A2}) = \emptyset$, $P_{A1} \cap P_{A2} \neq P_C$, $P_{R1} \cap P_{R2} = P_C$ and $T_1 \cap T_2 \neq \emptyset$. Then, the net $N = (\{p^0\} \cup P_A \cup P_R, T, F)$ is also an S^3PR resulting from the composition of N_1 and N_2 via the set of common P_C and defined as follows: (1) $p^0 = \{p^0{}_1\} \cup \{p^0{}_2\}$, $P_R = P_{R1} \cup P_{R2}$, $P_A = P_{A1} \cup P_{A2}$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$.*

The composition of n S^2PR N_1-N_n via P_C, is denoted by $\bigotimes_{i=1}^{n} N_i$. $\overline{N_i}$ is used to denote the S^2P from which the S^2PR N_i is formed.

Definition 5 [23]. *Let $N = (\{p^0\} \cup P_A \cup P_R, T, F)$ be an S^3PR. M_o is an initial marking of N. (N, M_o) is said to be an acceptably marked S^3PR if (1) (N, M_o) is an acceptably marked S^2PR, (2) $N = N_1 \circ N_2$, where (N_i, M_{oi}) is said to be an acceptably marked S^3PR and*

- $\forall p \in P_{Ai} \cup \{p^0{}_i\}$, $M_o\,(p) = M_{oi}\,(p)$, $\forall i \in \{1,2\}$.
- $\forall r \in P_{Ri} \smallsetminus P_C$, $M_o(r) = M_{oi}\,(r)$, $\forall i \in \{1,2\}$.
- $\forall r \in P_{Ri}$, $M_o\,(p) = max\,\{M_{o1}\,(r), M_{o2}\,(r)\}$, $\forall i \in \{1,2\}$.

Definition 6 [23]. *Let N be an S^3PR, A non-empty set $S \subseteq P$ is said to be a siphon in N if $\cdot S \subseteq S\cdot$. When a siphon does not include other siphons, it is said to be a minimal siphon.*

Definition 7 [24]. *Let S be a minimal siphon in an S^3PR N. A minimal siphon S is said to be strict if $\cdot S \subsetneq S \cdot$. Let $\Pi = \{S_1, S_2, S_3, \ldots , S_k\}$ be a set of SMSs of N. We have $S = S_A \cup S_R$, $S_R = S \cap P_R$, and $S_A = S \setminus S_R$, where S_A denotes the places of operations and S_R denotes the places of resources.*

Definition 8 [23]. *Let $r \in P_R$ be a reliable resource place in an S^3PR N. The operation places that use r are known as the set of holders of r, indicated by $H(r) = \{p\backslash p \in P_A, p \in \cdot r \cap P_A \neq \emptyset\}$. $[S]$ is said to be the complementary set of S if $[S] = (U_{r\in S_R} H(r)) \smallsetminus SA$.*

Definition 9 [24]. *Let (N_a, M_a) and (N_b, M_b) be marked Petri nets; $N_i = (P_i, T_i, F_i, W_i)$, where $i = a, b$. We call (N, M) with $N = (P, T, F, W)$ a synchronous net resulting from the integration of (N_a, M_a) and (N_b, M_b) and denote it as $(N_a, M_a) \| (N_b, M_b)$ when the following conditions are satisfied: (1) $P = P_a \cup P_b$, and $P_a \cap P_b = \emptyset$. (2) $T = T_a \cup T_b$. (3) $F = F_a \cup F_b$. (4) $W\,(e) = W_i\,(e)$, where $e \in F_i$, $i= a, b$. (5) $M(p) = M_i\,(p)$, where $p \in P_i$, $i = a, b$.*

Definition 10 [25]. *Let (N, M_o) be an S^3PR with $N = (P_A \cup \{p^0\} \cup P_R, T, F, M_o)$. The deadlock controller for (N, M_o) developed by Ezpeleta et al. [1] is denoted as $(V, M_{Vo}) = (P_V, T_V, F_V, M_{Vo})$, where (1) $P_V = \{V_S \setminus S \in \Pi\}$ is a set of control places. (2) $T_V = \{t \setminus t \in \cdot V_S \cup V_S \cdot\}$. (3) $F_V \subseteq (P_V \times T_V) \cup (T_V \times P_V)$ is the set of directed arcs that join the control places with transitions (and vice versa). (4) For all $V_S \in P_V$, $M_{Vo}\,(V_S) = M_{Vo}\,(S) - 1$, where $M_{Vo}\,(V_S)$ is called an initial marking of a control place V_S.*

We call (N_V, M_{Vo}) a controlled Petri net model resulting from the integration of (V, M_{Vo}) and (N, M_o), denoted as $(V, M_{Vo}) \| (N, M_o)$. A control place or monitor is inserted to each SMS to ensure the liveness of a Petri net, and all SMSs can never be unmarked. The proposed policy is simple and guarantees success. However, it leads to a more complex Petri-net-controlled system than the original Petri net model, because the number of added monitors is equal to that of the SMSs in the target Petri net model, and the number of associated arcs is larger than that of the added monitors. According to the strict minimal siphon concept, the developed deadlock prevention approach proposed by [1] is described by Algorithm 1.

Algorithm 1: Strict minimal siphon-based algorithm

Input: Original S^3PR Petri net model (N, M_o)
Output: Controlled net (N_V, M_{Vo}).
Step 1: Compute the set of SMS Π for N.
Step 2: for each $S \in \Pi$ do

- Add a control place V_S. /* By using Definition 10.*/
- Add Vs output arcs; all arc weights are unitary. /* By using Definition 10.*/
- Add Vs input arcs; all arc weights are unitary. /* By using Definition 10.*/
- Compute $M_{Vo}(V_S)$. /* By using Definition 10.*/

    end for
Step 3: Output a controlled net (N_V, M_{Vo}).
Step 4: End

Consider the S^3PR Petri net model shown in Figure 1. The Petri net model involves eleven places and eight transitions. The places can be described as the following set partition: $P^0 = \{p_1, p_8\}$, $P_R = \{p_9, p_{10}, p_{11}\}$, and $P_A = \{p_2, p_3, \dots, p_7\}$. The model has 20 reachable markings and eight minimal siphons, three of which are SMSs. The siphons are $S_1 = \{p_4, p_7, p_9, p_{10}, p_{11}\}$, $S_2 = \{p_4, p_6, p_{10}, p_{11}\}$, and $S_3 = \{p_3, p_7, p_9, p_{10}\}$. According to Definitions 2, 3, and 5

(1) For S_1: $S_A = \{p_4, p_7\}$, $S_R = \{p_9, p_{10}, p_{11}\}$, $H(p_9) = \{p_2, p_7\}$, $H(p_{10}) = \{p_3, p_6\}$, $H(p_{11}) = \{p_4, p_5\}$, $[S_1] = \{p_2, p_3, p_5, p_6\}$, $\cdot V_{S1} = \{t_3, t_7\}$, $V_{S1}\cdot = \{t_1, t_5\}$, and $M_{Vo}(V_{S1}) = 2$.

(2) For S_2: $S_A = \{p_4, p_6\}$, $S_R = \{p_{10}, p_{11}\}$, $H(p_{10}) = \{p_3, p_6\}$, $H(p_{11}) = \{p_4, p_5\}$, $[S_2] = \{p_3, p_5\}$, $\cdot V_{S2} = \{t_3, t_6\}$, $V_{S2}\cdot = \{t_2, t_5\}$, and $M_{Vo}(V_{S2}) = 1$.

(3) For S_3: $S_A = \{p_3, p_7\}$, $S_R = \{p_{10}, p_{11}\}$, $H(p_{10}) = \{p_3, p_6\}$, $H(p_{11}) = \{p_4, p_5\}$, $[S_3] = \{p_2, p_6\}$, $\cdot V_{S3} = \{t_2, t_7\}$, $V_{S3}\cdot = \{t_1, t_5\}$, and $M_{Vo}(V_{S3}) = 1$.
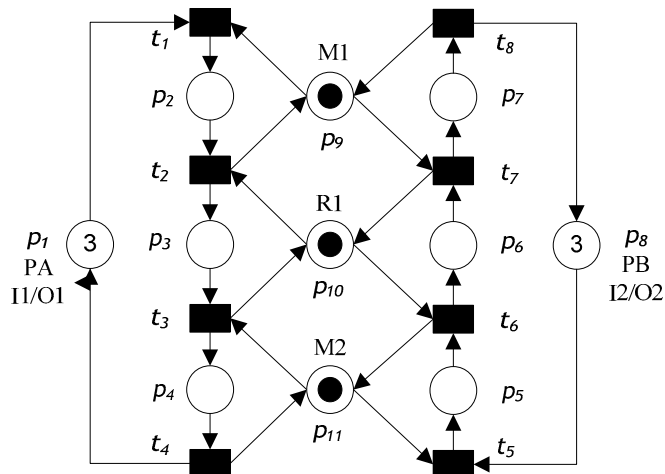

Figure 1. S^3PR Petri net model of an AMS.

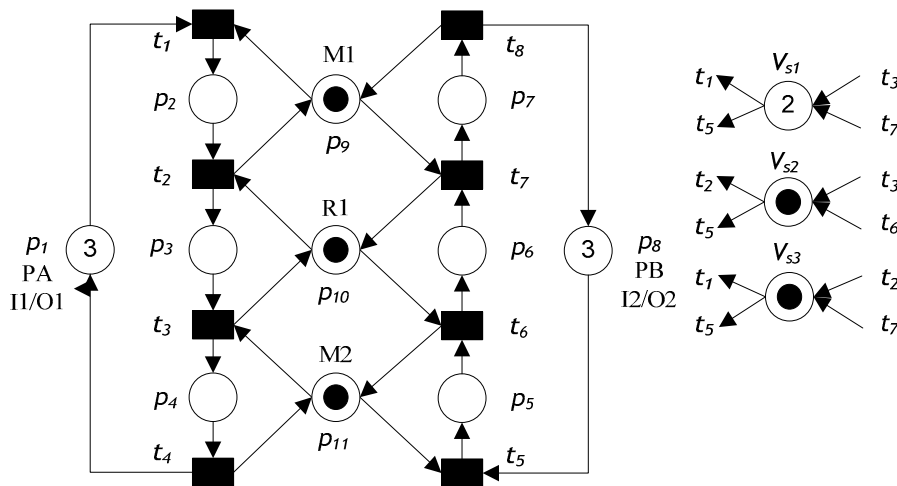After monitors have been added using Algorithm 1, we obtain the controlled Petri net model shown in Figure 2.

Figure 2. Controlled S^{3}PR Petri net model.

Definition 11. *Let (N, M_0) be an S^{3}PR with $N = (P_A \cup \{p^0\} \cup P_R, T, F, M_0)$. The deadlock controller for (N, M_0) developed in Ezpeleta et al. [1] is denoted as $(V, M_{V0}) = (P_V, T_V, F_V, M_{V0})$. Here, (V, M_{V0}) can be reduced and replaced by a colored common deadlock control subnet, which is a PN $N_{DC} = (\{p_{combined}\}, \{T_{DCi} \cup T_{DC0}\}, F_{DC}, C_{vsi})$, where $p_{combined}$ is called the merged control place of all monitors $P_V = \{V_S \setminus S \in \Pi\}$. $T_{DCi} = \{t \setminus t \in {}^\bullet V_S\}$. $T_{DC0} = \{t \setminus t \in V_S{}^\bullet\}$. $F_{DC} \subseteq (\{p_{combined}\} \times \{T_{DCi} \cup T_{DC0}\}) \cup (\{T_{DCi} \cup T_{DC0}\} \times \{p_{combined}\})$ is the set of arrows that join the merged control place with transitions (and vice versa). C_{cri} is the color that maps $p_{combined}$ into colors $C_{vsi} \in SC$, where $SC = \cup_{i \in Vs}\{C_{vsi}\}$. (N_{DC}, M_{DC0}) is called a colored common deadlock control subnet. For all $V_S \in P_V$, $M_{DC0}(p_{combined}) = \sum M_{V0}(V_S)$, where $M_{DC0}(p_{combined})$ is an initial token with the colors marking of the merged monitor.*

Figure 3 shows $p_{combined}$, the merged control place of all monitors P_V of the controlled Petri net model from Figure 2, according to Definition 6.
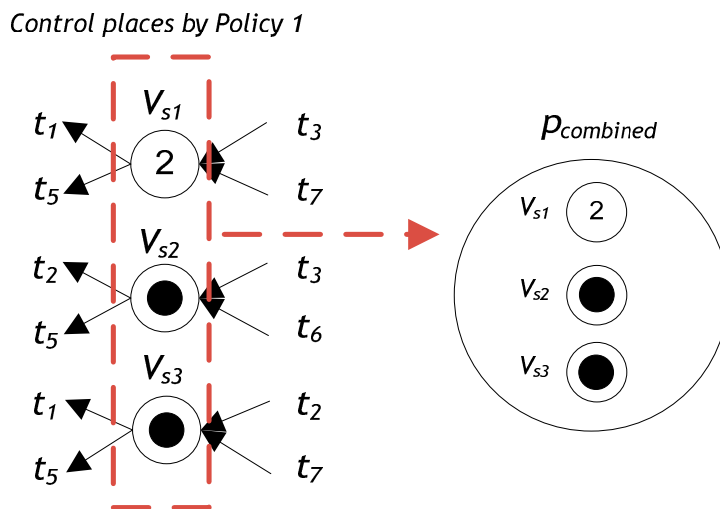

Figure 3. Merged control place for all monitors P_V.

The output arcs of $p_{combined}$ are connected to the source transitions T_{DC0}, which lead to the sink transitions of S. Transitions $V_{si}{}^\bullet$ for all monitors P_V augmented from Algorithm 1 are defined as $V_{S1}{}^\bullet = \{t_1, t_5\}$, $V_{S2}{}^\bullet = \{t_2, t_5\}$, and $V_{S3}{}^\bullet = \{t_1, t_5\}$. Thus, T_{DC0} can be denoted as $T_{DC0} = \cup_{i \in Vs}\{V_{Si}{}^\bullet\}$, so $T_{DC0} = \{2t_1, t_2, 3t_5\}$, as shown in Figure 4.
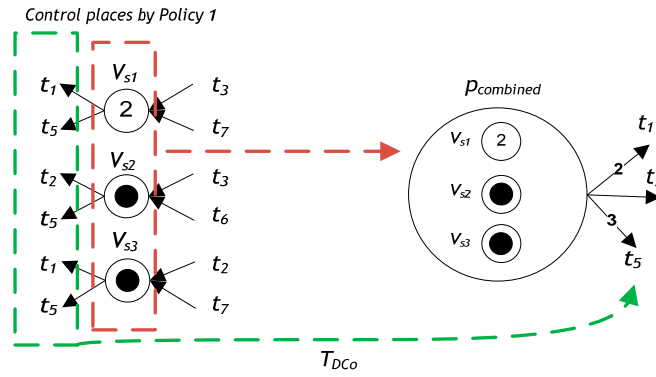
Figure 4. Output arcs of $p_{combined}$ for all monitors P_V.

The input arcs of $p_{combined}$ are connected with the output of S, denoted as T_{DCi}. Transitions $\cdot V_{si}$ for all monitors P_V augmented from Algorithm 1 are defined as $\cdot V_{s1} = \{t_3, t_7\}$, $\cdot V_{s2} = \{t_3, t_6\}$, and $\cdot V_{s3} = \{t_2, t_7\}$. Thus, T_{DCi} can be represented by $T_{DCi} = \cup_{i \in V_s}\{\cdot V_{si}\}$, so $T_{DCi} = \{t_2, 2t_3, t_6, 2t_7\}$, as shown in Figure 5.
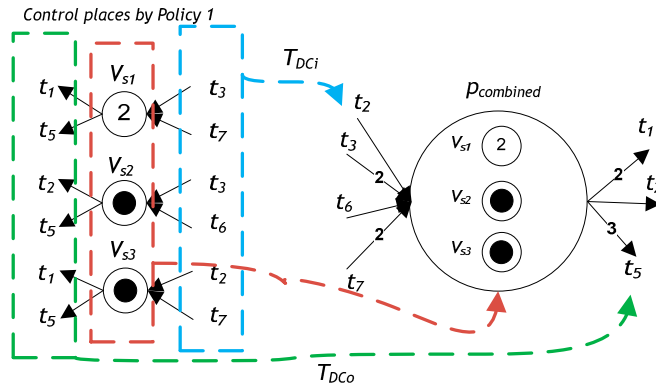
Figure 5. Output arcs of $p_{combined}$ for all monitors P_V.

Moreover, $M_{DCo}(p_{combined}) = \sum M_{Vo}(V_S) = M_{Vo}(V_{S1}) + M_{Vo}(V_{S2}) + M_{Vo}(V_{S3}) = 2 + 1 + 1 = 4$. Thus, in the model of the Petri net from Figure 2, we have three color types: $SC = \{C_{vs1}, C_{vs2}, C_{vs3}\}$. Therefore, the total number of colored tokens is 4: we have two tokens of C_{vs1} color, one token of C_{vs2} color, and one token of C_{vs3} color, as shown in Figure 6.
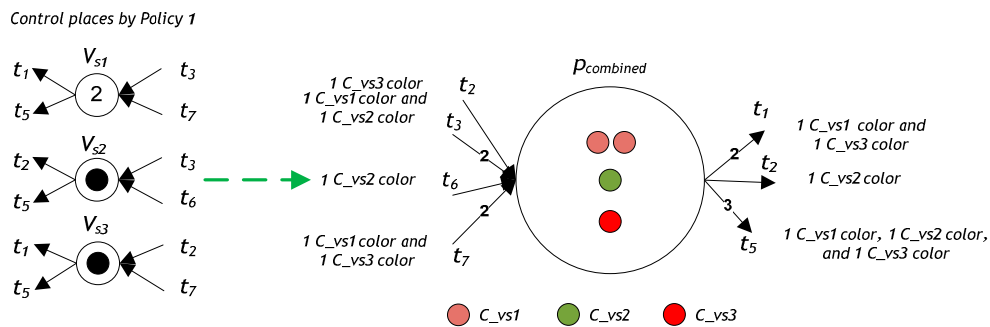

Figure 6. Merged controller for all monitors P_V.

Definition 12. *Let (N, M_o) be an S^3PR with $N = (P_A \cup \{p^0\} \cup P_R, T, F, M_o)$ and (N_{DC}, M_{DCo}) a deadlock controller for (N, M_o) created by Definition 11 with $N_{DC} = \{p_{combined}\}, \{T_{DCi} \cup T_{DCo}\}, F_{DC}, C_{vsi}, M_{DCo})$. We call (N_C, M_{Co}) a controlled marked Petri net, denoted as $(N_C, M_{Co}) = (N, M_o) \parallel (N_{DC}, M_{DCo})$ and called the composition of (N, M_o) and (N_{DC}, M_{DCo}), where $N_C = (P_A \cup \{p^0\} \cup P_R \cup \{p_{combined}\}, T \cup T_{DCi} \cup T_{DCo}, F \cup F_{DC}, C_R, M_{Co})$ be a colored controlled marked S^3PR, and $R(N_C, M_{Co})$ be its reachable graph.*

Theorem 1. *The colored controlled net (N_C, M_{Co}) is live.*

Proof. We must prove that all transitions T, T_{DCi}, T_{DCo} in (N_C, M_{Co}) are live. No strict minimal siphon is emptied. In addition, no new strict minimal siphon is created, since all $t_1 \in T$ are live. For all $t_2 \in T_{DCi}$; if $\forall p_i \in \bullet\, t_2$, $M_C(p_i) > 0$, then t_2 can fire in any case because it is uncontrollable. Thus, M_C ($p_{combined}$) > 0, for all $t_3 \in T_{DCo}$, if M_C ($p_{combined}$) > 0, then t_3 can fire. Therefore, controlled net (N_C, M_{Co}) is live.　□

According to the concepts of SMSs and colored Petri nets, the developed policy is stated in Algorithm 2.

Algorithm 2: Integrated Strict Minimal Siphon And Colored Petri Nets-Based Algorithm

---

Input: Petri net models (N, M_o) and (V, M_{Vo}).
Output: Colored controlled S³PR Petri net (N_C, M_{Co}).
Step 1: Combine all monitors P_V into one monitor ($p_{combined}$), considering the following steps:

1.　Add $p_{combined}$ output arcs T_{DCo}. /* By Definition 11.*/
2.　Add $p_{combined}$ input arcs T_{DCi}. /* By Definition 11.*/
3.　Define colors for monitors P_V /* By Definition 11.*/
4.　Compute M_{DCo} ($p_{combined}$) $= \sum M_{Vo}$ (V_S), where M_{DCo} ($p_{combined}$) is an initial token with the colors marking of a merged monitor. /* By Definition 11.*/

Step 2: Insert the combined monitor into the Petri net model (N, M_o). The obtained net is denoted as (N_C, M_{Co}).
Step 3: Output (N_C, M_{Co}).
Step 4: End

Figure 7 shows the proposed single controller for the controlled Petri net model from Figure 2 by using Algorithm 2. In the net shown in Figure 7, when transition t_1 fires, the system selects only one token from input place p_1, one token from resource place p_9, one token of color C_{vs1} from $p_{combined}$, and one token of color C_{vs3} from $p_{combined}$, and it transfers them into p_2. Moreover, when transition t_2 fires, the system selects only one token from operation place p_2, one token from resource place p_{10}, and one token of color C_{vs2} from $p_{combined}$ and transfers them into p_3. When transition t_5 fires, the system selects only one token from input place p_8, one token from resource place p_{11}, one token of color C_{vs2} from $p_{combined}$, one token of color C_{vs2} from $p_{combined}$, and one token of color C_{vs3} from $p_{combined}$ and transfers them into p_5. When transition t_2 fires, it creates a color C_{vs3} on the tokens from p_2 and p_{10} and transfers them into common place $p_{combined}$. In addition, when the transition t_3 fires, it creates two colors C_{vs1} and C_{vs2} on the tokens from p_3 and p_{11} and transfers them into place $p_{combined}$. Moreover, when transition t_6 fires, the system creates a color C_{vs2} on the tokens from p_5 and p_{10} and transfers them into place $p_{combined}$. Finally, when transition t_7 fires, the system creates two colors C_{vs1} and C_{vs3} on the tokens from p_6 and p_9 and transfers them into $p_{combined}$.

By default, colors are inherited: when a transition T_{DCo} fires, the system gathers all colored tokens from the input place $p_{combined}$ and then transfers these colored tokens to the output place p_i. However, color inheritance can be prohibited by overriding.

After implementing a Petri net model shown in Figure 7 by using the GPenSIM code, we usually obtain three files: the Petri net definition file (PDF), common processor file (COMMON_PRE file), and main simulation file (MSF). Figure 8 shows the resulting PDF file and defines the static Petri net model by declaring the sets of places, transitions, and arcs. Figure 9 shows the MSF file used to compute the simulation results. Figure 10 displays the resulting MSF COMMON_PRE file and defines the conditions for the enabled failure and recovery transitions to start firing based on the colored tokens, mean time to failure, and mean time to repair.
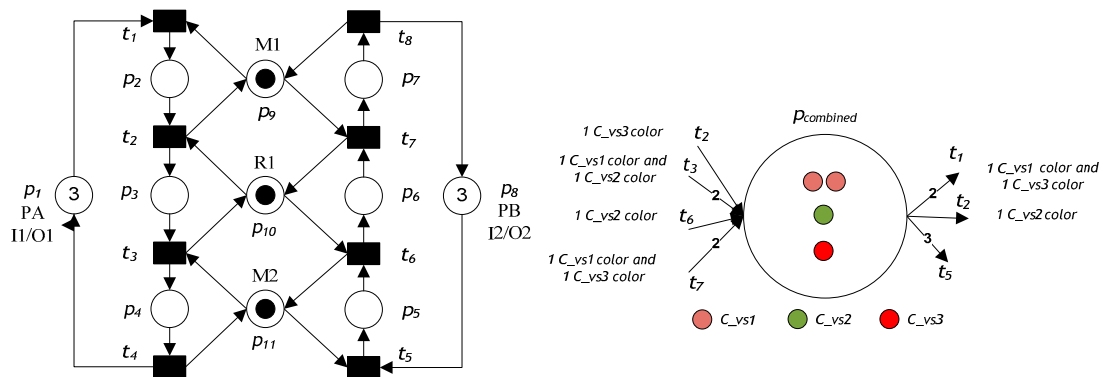
Figure 7. Colored controlled S^{3}PR Petri net model Algorithm 2.

Figure 8. PDF file of the colored controlled S^{3}PR Petri net model from Figure 7.

Figure 9. Part of the MSF file of the colored controlled S^{3}PR Petri net model from Figure 7.

```matlab
 1   function [fire, transition] = COMMON_PRE (transition)
 2       global global_info;
 3       % ===================Start combined Places======================
 4       % ===================t1======================
 5       if strcmp(transition.name, 't1'),
 6           tokID1 = tokenEXColor('pcombined',1,{'C_vs1'});
 7           tokID2 = tokenEXColor('pcombined',1,{'C_vs3'});
 8           selected_tokens = [tokID1 tokID2];    % tokens to be removed
 9           fire = all(selected_tokens); % must have both "vs3" and "vs4"
10           transition.selected_tokens = selected_tokens;  % tokens to be removed
11           transition.override = 1; % only sum as color – NO inheritance
12               return;
13       end;
14       %===================t2======================
15       if strcmp(transition.name, 't2'),
16           tokID2 = tokenEXColor('pcombined',1,{'C_vs2'});
17           if tokID2 >0,
18            transition.selected_tokens = tokID2;   % tokens to be removed
19            fire = 1;
20            transition.override = 1; % only sum as color – NO inheritance
21       else
22            fire = 0;
```

Figure 10. Part of the COMMON_PRE file of the colored controlled S^3PR Petri net model from Figure 7.

To make the work more solid and well-positioned, we have developed a Trust-based colored controlled Petri net (TCCPN) [26–29] and comparing the proposed work with relevant TCCPNs.

Definition 13. *Let* (N_{TM}, M_{TMo}) *be a Trust-based colored controlled Petri net (TCCPN)* S^3PR *with* $N_{TM} = (P_A \cup \{p^0\} \cup P_R \cup \{p_{combined}\}, T \cup T_{DCi} \cup T_{DCo}, F \cup F_{DC}, C_R, \eta, \tau, \psi, M_{TMo})$, *and* R (N_{TM}, M_{TMo}) *be its reachable graph, where*

1. *η is the arcs weight, which denotes the importance or probability of input arcs into a transition. If there is an arc (p, t), η $(p, t) = c$ indicates there is a probability of η (p, t) encouraging the token entering t from p. If the token has a capacity h, the new capacity will be $h * c$.*
2. *τ is a time guard for transition $t \in (T \cup T_{DCi} \cup T_{DCo})$, τ: $t \rightarrow [e, f]$, τ (t) indicates transition t can only fire during e and f. Particularly, if $e = f$, that indicates the transition can only occur through e.*
3. *ψ is the threshold of token in $p \in (P_A \cup \{p^0\} \cup P_R \cup \{p_{combined}\})$, ψ: $p \rightarrow R$, and R is a real type data. Ψ $(p) = r_1$, indicates when the number of tokens in p is less than or equal to r_1, p can reach a new position.*

To model a TCCPN, there are many types of factors that can have an influence on trust in colored controlled net S^3PR, and a non-negative real number can represent the value of each factor type. An assessment process will consume factors for aggregating a new trust value. We use E^{in} to characterize the input factors consumed and use E^{out} to characterize the aggregation trust value. For an assessment process AP_k, $E^{in}(AP_k)$ is related to the input place and E^{ou} (AP_k) is related to the output place. There are rules for firing transitions in a TCCPN and are stated as below:

Definition 14. *Let* (N_{TM}, M_{TMo}) *be a Trust-based colored controlled Petri net (TCCPN)* S^3PR *with* $N_{TM} = (P_A \cup \{p^0\} \cup P_R \cup \{p_{combined}\}, T \cup T_{DCi} \cup T_{DCo}, F \cup F_{DC}, C_R, \eta, \tau, \psi, M_{TMo})$, *a transition* T_k *under the marking M can be fired when the following conditions are satisfied:*

1. $t \in \tau(T_k)$
2. $E^{in}(AP_k) > 0$
3. $E^{out}(AP_k) \geq \Psi(p_i)$
4. *for all* $p \in \cdot t$, $M(p) \geq W$ (p, t)

where $\tau(t)$ is the valid firing time in TCCPN, and E^{in} (AP_k) is the current value of the token in the input place. E^{out} (AP_k) is the current value of the token in the output place, and Ψ (p_i) is the threshold for entering p_i. Note that an assessment process AP_k may have more than one input place and output place.

In addition, there are rules for new markings, when a transition t fires, a token value can be changed and it will be kept in a new place because of the threshold, $E^{out}(AP_k) = \sum_{i=1}^{nf} \eta_i * E^{in}_i(AP_k)$, where nf is the number of input factors. Then, the new marking will be changed as $M'(p) = M(p) - W(p, t) + W(t, p)$. In order to demonstrate the TCCPN, reconsider Figure 7, it describes a system as follows:

1. $\tau(t)$: $\tau(t_1) = 0$, $\tau(t_2) = 3$, $\tau(t_3) = 2$, $\tau(t_4) = 4$, $\tau(t_5) = 0$, $\tau(t_6) = 4$, $\tau(t_7) = 5$, $\tau(t_8) = 3$;
2. $M_o = (p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_8\ p_9\ p_{10}\ p_{11}\ p_{combined}) = (3\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 1\ 1\ 1\ 4)$;
3. Assessment process is a combined place $Ap_{combined}$
4. The input assessment process is $T_k = \{t_2, 2t_3, t_6, 2t_7\}$
5. The importance on arcs: $\eta\ (p_2, t_2) = 0.5$, $\eta(p_{10}, t_2) = 0.5$, $\eta(p_3, t_3) = 0.5$, $\eta(p_{11}, t_3) = 0.5$, $\eta(p_5, t_6) = 0.5$, $\eta(p_{10}, t_5) = 0.5$, $\eta(p_6, t_7) = 0.5$, $\eta(p_9, t_7) = 0.5$
6. The factors' values of E^{in} ($Ap_{combined}$): if t_2 enabled, $E^{in}_1(Ap_{combined}) = (1, 1)$, if t_3 enabled, $E^{in}_2(Ap_{combined}) = (2, 2)$, if t_6 enabled, $E^{in}_3(Ap_{combined}) = (1, 1)$, if t_7 enabled, $E^{in}_4(Ap_{combined}) = (2, 2)$.
7. $\Psi(p)$: $\Psi(p_1) = 0$, $\Psi(p_2) = 0$, $\Psi(p_3) = 0$, $\Psi(p_4) = 0$, $\Psi(p_5) = 0$, $\Psi(p_6) = 0$, $\Psi(p_7) = 0$, $\Psi(p_8) = 0$, $\Psi(p_9) = 0$, $\Psi(p_{10}) = 0$, $\Psi(p_{11}) = 0$, and Ψ ($p_{combined}$) ≤ 4

Places $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}$ and p_{11} obtain tokens unconditionally, their thresholds are set as zero. t_2, t_3, t_6, t_7 represent the four transitions that can fire. If any of four transitions fires, new tokens with colors are built. For example, when transition t_2 fires, it creates a color C_{vs3} on the tokens from p_2 and p_{10} and transfers them into common place $p_{combined}$. For transition t_2, there are two factors: p_2 and p_{10}

$E^{out}(Ap_{combined}) = \sum_{i=1}^{nf} \eta_i * E^{in}_i (Ap_{combined}) = \eta_1 * E^{in}_1(Ap_{combined}) + \eta_2 * E^{in}_2(Ap_{combined}) = 0.5*1 + 0.5*1 = 1$. Since if the current tokens in $p_{combined}$ place + 1 ≤ 4, $p_{combined}$ can be reached.

If transition t_3 fires, it creates colors C_{vs1} and C_{vs2} on the tokens from p_3 and p_{11} and transfers them into common place $p_{combined}$. For transition t_3, there are two factors: p_3 and p_{11}

$E^{out}(Ap_{combined}) = \sum_{i=1}^{nf} \eta_i * E^{in}_i (Ap_{combined}) = \eta_1 * E^{in}_1(Ap_{combined}) + \eta_2 * E^{in}_2(Ap_{combined}) = 0.5*2 + 0.5*2 = 2$. Since if the current tokens in $p_{combined}$ place + 2 ≤ 4 $p_{combined}$ can be reached.

When transition t_6 fires, it creates a color C_{vs2} on the tokens from p_5 and p_{10} and transfers them into common place $p_{combined}$. For transition t_6, there are two factors: p_5 and p_{10}

$E^{out}(Ap_{combined}) = \sum_{i=1}^{nf} \eta_i * E^{in}_i(Ap_{combined}) = \eta_1 * E^{in}_1(Ap_{combined}) + \eta_2 * E^{in}_2(Ap_{combined}) = 0.5*1 + 0.5*1 = 1$. Since if the current tokens in $p_{combined}$ place + 1 ≤ 4 $p_{combined}$ can be reached.

If transition t_7 fires, it creates colors C_{vs1} and C_{vs3} on the tokens from p_6 and p_9 and transfers them into common place $p_{combined}$. For transition t_7, there are two factors: p_6 and p_9

$E^{out}(Ap_{combined}) = \sum_{i=1}^{nf} \eta_i * E^{in}_i(Ap_{combined}) = \eta_1 * E^{in}_1(Ap_{combined}) + \eta_2 * E^{in}_2(Ap_{combined}) = 0.5*2 + 0.5*2 = 2$. Since if the current tokens in $p_{combined}$ place + 2 ≤ 4 $p_{combined}$ can be reached.

Finally, comparing the proposed work with a relevant developed TCCPN is shown in Section 4.

4. Case Study

In this section, we show the results of the experiments with the proposed approach. Specifically, we use an AMS example available in the literature: the AMS Petri net model given in Piroddi et al. [30], Chen et al. [8], Chen and Li [31], Chen et al. [32], and TCCPN [26–29]. The Petri net model is displayed in Figure 11; it includes 14 transitions and 19 places. The places can be described as the following set partition: $P^0 = \{p_1, p_{19}\}$, $P_R = \{p_{13}, \dots, p_{18}\}$, and $P_A = \{p_2, \dots, p_{14}\}$. The properties of the developed Petri net models are obtained using the free GPenSIM tool [22]. We find that it has 282 reachable markings, and the system is not live (it has a deadlock).
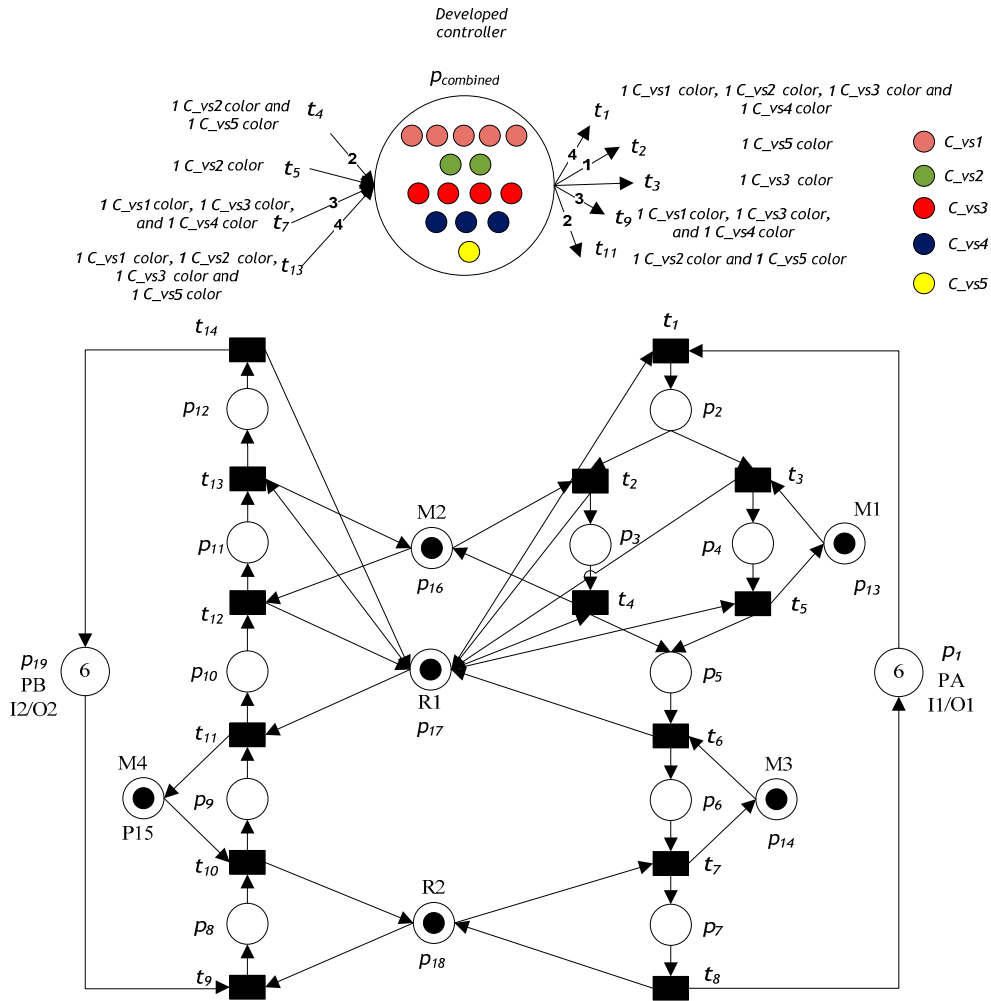
Figure 11. Colored controlled Petri net model of the system.

We apply the proposed deadlock-prevention algorithm to this case study. Without considering recovery subnets, the system model has five SMSs that may be empty: $S_1 = \{p_7, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}\}$, $S_2 = \{p_5, p_{12}, p_{13}, p_{16}, p_{17}\}$, $S_3 = \{p_2, p_7, p_{12}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}\}$, $S_4 = \{p_2, p_7, p_{10}, p_{12}, p_{14}, p_{15}, p_{17}, p_{18}\}$, and $S_5 = \{p_2, p_5, p_{12}, p_{16}, p_{17}\}$. Based on the suggested deadlock-prevention algorithm (Algorithm 1), five monitors are inserted to protect the five SMSs from being emptied. The required control places using Algorithm 1 are designed as follows:

(1)　$^\bullet V_{S1} = \{t_7, t_{13}\}$, $V_{S1}^\bullet = \{t_1, t_9\}$, and $M_{V0}(V_{S1}) = 5$.
(2)　$^\bullet V_{S2} = \{t_4, t_5, t_{13}\}$, $V_{S2}^\bullet = \{t_1, t_{11}\}$, and $M_{V0}(V_{S2}) = 2$.
(3)　$^\bullet V_{S3} = \{t_7, t_{13}\}$, $V_{S3}^\bullet = \{t_1, t_9\}$, and $M_{V0}(V_{S3}) = 4$.
(4)　$^\bullet V_{S4} = \{t_7, t_{11}\}$, $V_{S4}^\bullet = \{t_1, t_9\}$, and $M_{V0}(V_{S4}) = 3$.
(5)　$^\bullet V_{S5} = \{t_4, t_{13}\}$, $V_{S5}^\bullet = \{t_2, t_{11}\}$, and $M_{V0}(V_{S5}) = 3$.

By Definition 11, a deadlock control subnet of the Petri net model illustrated in Figure 11 is $N_{DC} = (p_{combined}, \{T_{DCi}, T_{DCo}\}, F_{DC}, C_{vsi})$, where $T_{DCo} = \{4t_1, t_2, t_3, 3t_9, 2t_{11}\}$, and $T_{DCi} = \{2t_4, t_5, 3t_7, 4t_{13}\}$. The initial token with a color marking of a combined monitor is $M_{DCo}(p_{combined}) = \sum M_{V0}(V_S) = M_{V0}(V_{S1}) + M_{V0}(V_{S2}) + M_{V0}(V_{S3}) + M_{V0}(V_{S4}) + M_{V0}(V_{S5}) = 5 + 2 + 4 + 3 + 1 = 15$ tokens. Thus, in the Petri net model illustrated in Figure 11, there are five color types, which are $SC = \{C_{vs1}, C_{vs2}, C_{vs3}, C_{vs4}, C_{vs5}\}$. Therefore, the total number of colored tokens is 15: five tokens of color C_{vs1}, two tokens of color C_{vs2}, four tokens of color C_{vs3}, three tokens of color C_{vs4}, and one token of color C_{vs5}, as shown in Figure 11.

In the net displayed in Figure 11, when transition t_1 fires, the system selects only one token from input place p_1, one token from resource place p_{17}. Additionally, the system selects tokens from $p_{combined}$: one token of color C_{vs1}, one token of color C_{vs2}, one token of color C_{vs3}, and one token of color C_{vs4}. If transition t_2 is fired, the system selects only one token from place p_2, one token from resource place p_{16}, and one token of color C_{vs5} from $p_{combined}$. Moreover, when transition t_3 fires, the system selects only one token from input place p_2, one token from resource place p_{13}, and one token of color C_{vs3} from $p_{combined}$.

If transition t_9 fires, the system selects only one token from input place p_{19}, one token from resource place p_{18}, one token of color C_{vs1} from $p_{combined}$, one token of color C_{vs3} from $p_{combined}$, and one token of color C_{vs4} from $p_{combined}$. In addition, when transition t_{11} fires, the system selects only one token from input place p_9, one token from resource place p_{15}, one token from resource place p_{17}, one token of color C_{vs2} from $p_{combined}$, and one token of color C_{vs5} from $p_{combined}$.

When transition t_4 fires, the system creates two colored tokens—one of color C_{vs2} and one of color C_{vs5}—and transfers them into the common place $p_{combined}$. Moreover, when transition t_5 fires, the system adds color C_{vs2} to the tokens and transfers them into the common place $p_{combined}$. If transition t_7 fires, the system creates three colored tokens—one of color C_{vs1}, one of color C_{vs3}, and one of color C_{vs4}—and transfers them into the common place $p_{combined}$. Finally, when transition t_{13} fires, the system creates four colored tokens—one of color C_{vs1}, one of color C_{vs2}, one of color C_{vs3}, and one of color C_{vs5}—and transfers them into the common place $p_{combined}$.

To test and validate the developed GPenSIM code, we compared it with the methods in Piroddi et al. [30], Chen et al. [8], Chen and Li [31], Chen et al. [32], and TCCPN [26–29]. The simulation was undertaken for 480 min. After running and simulating the Petri net model in MATLAB, we obtained the results summarized in Tables 2 and 3. Table 2 shows the results in terms of the number of monitors, number of arcs, liveness, and reachable marking. We observe that the proposed approach provides a supervisor with only a single control place and 9 arcs, both of which are minimal compared with other techniques in Piroddi et al. [30], Chen et al. [8], Chen and Li [31], and Chen et al. [32]. Table 3 displays the results in terms of utilization of the robots and machines, throughput of Part A and Part B, work-in-process (WIP), and total time in system (throughput time). In terms of the resource utilization, all methods obtain approximately the same values, as shown in Figure 12. Moreover, from the viewpoint throughput, the proposed method can provide greater throughput than other techniques as shown in Figure 13. In term of WIP, the proposed method leads to better WIP than the other techniques as shown in Figure 14. With respect to throughput time of Part A and Part B, overall, the proposed method can obtain less throughput time than other techniques as shown in Figures 15 and 16. Therefore, the proposed method is valid, it can give sufficiently accurate results, and it can potentially be applied to other cases.

Table 2. Comparison with the existing policies: number of monitors, number of arcs, liveness, and reachable marking.

Parameters	Chen et al. [8]	Piroddi et al. [30]	Chen and Li [31]	Chen et al. [32]	TCCPN [26–29]	Proposed Method
Monitors	8	5	2	2	1	1
Arcs	37	23	12	12	9	9
Liveness	Live	Live	Live	Live	Live	Live
Reachable marking	205	205	205	205	205	205

Table 3. Comparison with the existing policies: utilization throughput, work-in-process, and throughput time.

Parameter	Chen et al. [8]	Piroddi et al. [30]	Chen and Li [31]	Chen et al. [32]	TCCPN [26–29]	Proposed Method
M 1 utilization%	18.75	17.7083	17.7083	17.7083	17.7083	17.7083
M 2 utilization%	35	33.3333	33.3333	33.3333	34.5833	33.9583
M 3 utilization%	12.5	13.75	14.375	14.375	11.875	12.5
M 4 utilization%	22.5	21.6667	20.8333	20.8333	23.3333	22.5
R 1 utilization%	39.5833	40	40.4167	40.4167	39.1666	39.58333
R 2 utilization%	29.375	30	30	30	30	30
Throughput of Part A (unit)	20	21	20	20	19	24
Throughput of Part B (unit)	26	25	26	26	27	23
Work-In-Process	3.9271	3.93331	3.8480	3.9667	3.4938	3.3854
Throughput time of Part A (min)	23.9500	22.8571	24	23.9235	25.2105	19.9583
Throughput time of Part B (min)	18.4230	19.200	18.3321	18.4615	17.7407	20.8260
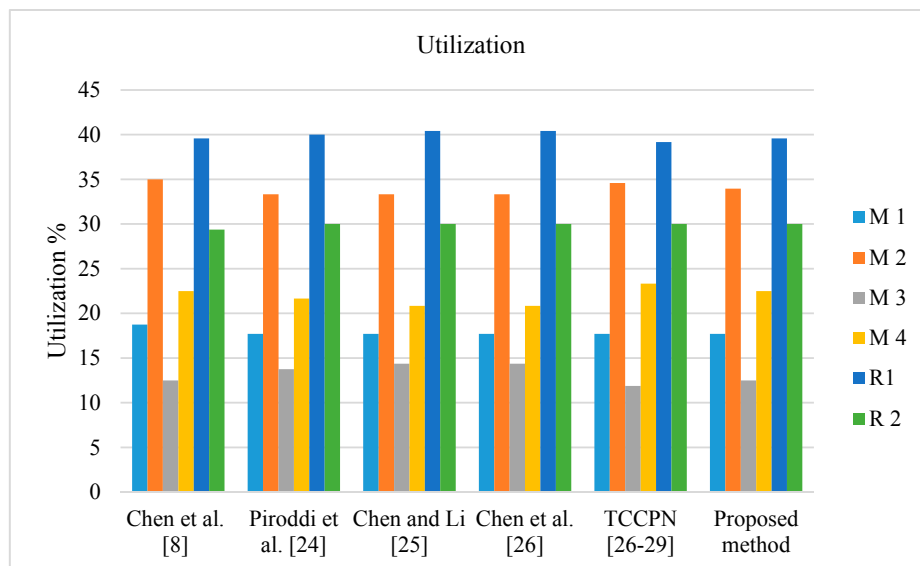
Figure 12. Comparison of utilization for the Petri net model from Figure 11.
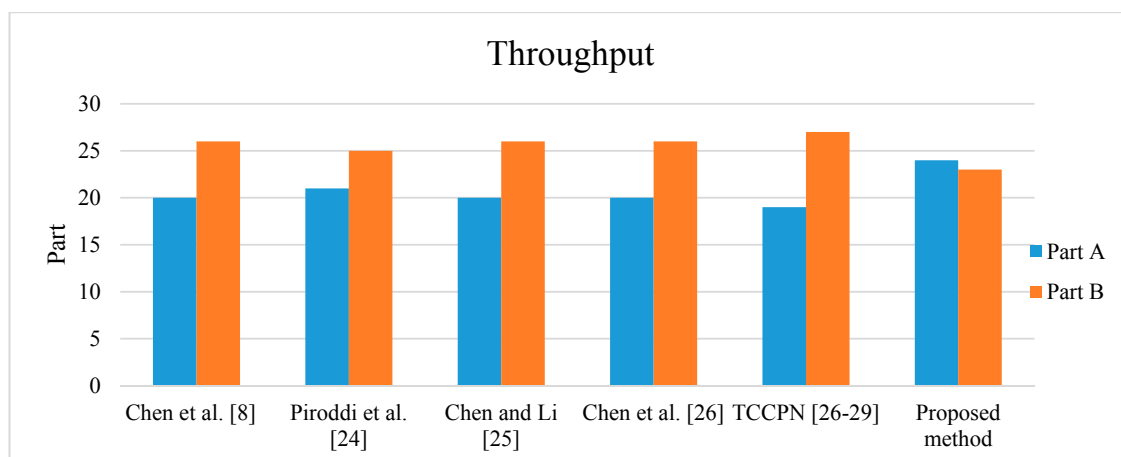
Figure 13. Comparison of throughput for the Petri net model from Figure 11.
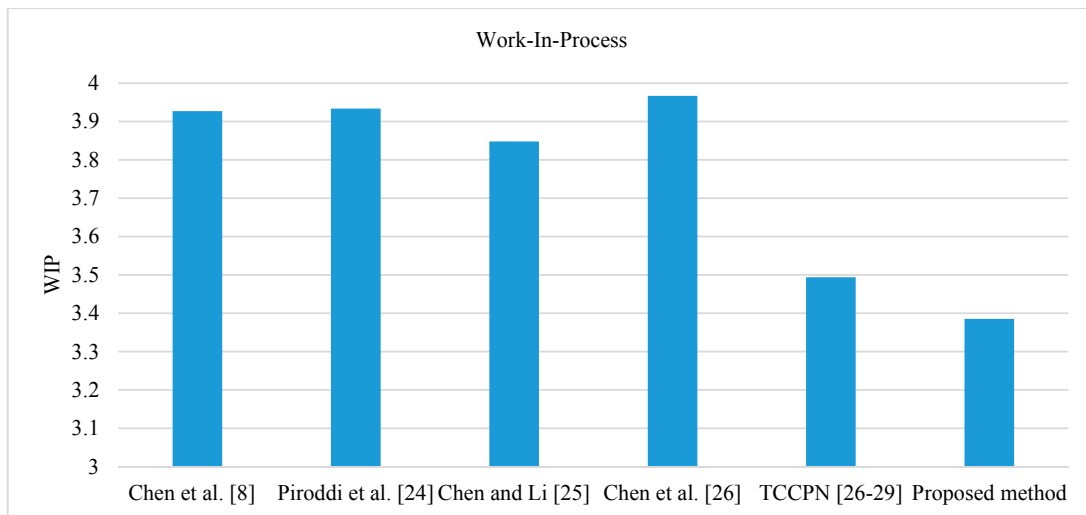
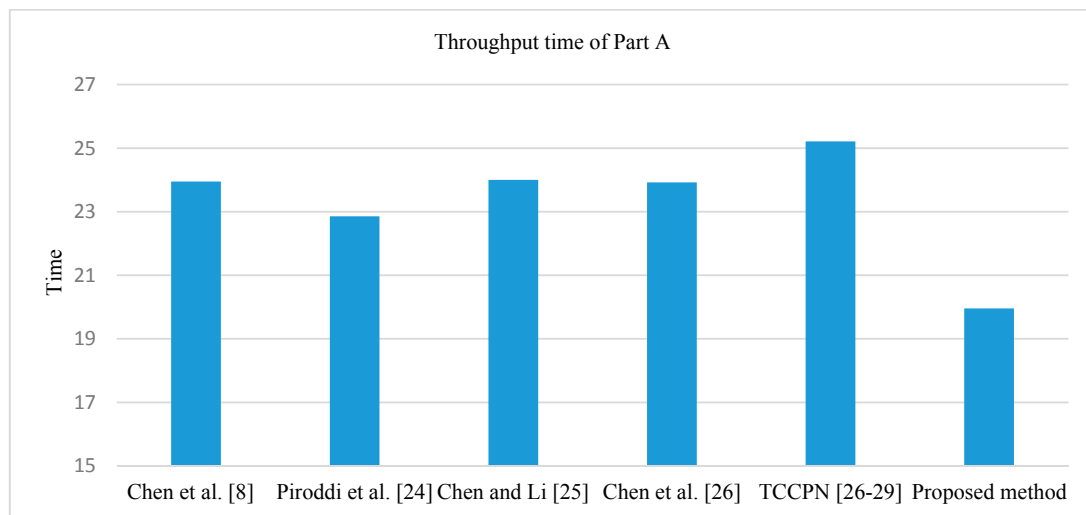**Figure 14.** Comparison of work-in-process for the Petri net model from Figure 11.

Figure 15. Comparison of throughput time of Part A for the Petri net model from Figure 11.
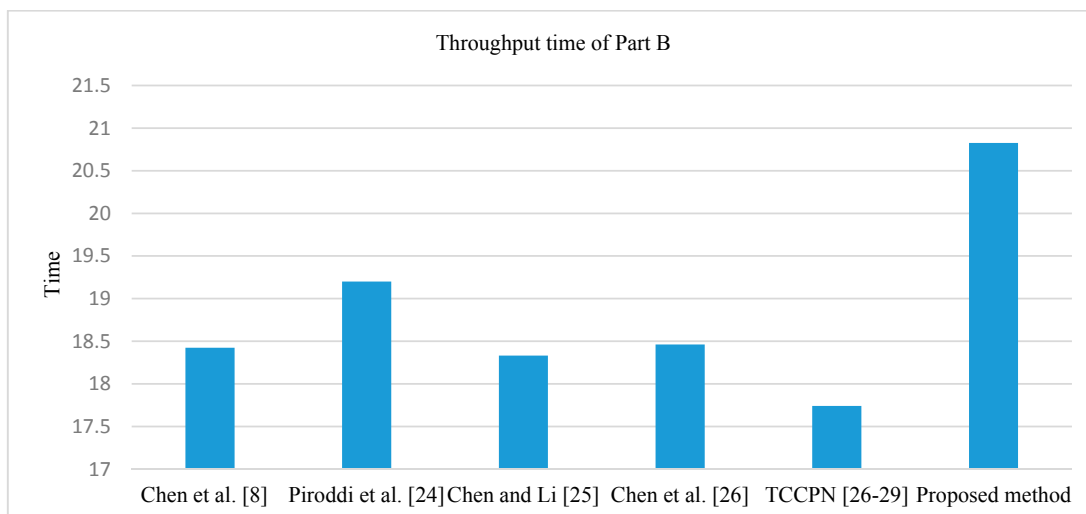

Figure 16. Comparison of throughput time of Part B for the Petri net model from Figure 11.

5. Conclusions

In this paper, we introduce a two-step controlled deadlock policy. In the first step, we create a Petri net controlled model using the deadlock prevention method based on SMSs proposed by [1]. In the second step, all control places obtained after the first step are merged into a single control place based on the colored Petri nets to mark all SMSs. We compare the proposed method with the methods of Piroddi et al. [30], Chen et al. [8], Chen and Li [31], and Chen et al. [32], and TCCPN [26–29]. According to our results, the proposed controller is more powerful, has a simpler structure, and does not need to calculate reachability graphs; therefore, it has low-overhead computation. The most challenging research topic in the future is that the controlled system that developed by previous deadlock control approaches may undergo changes of control requirements and specifications such as:

1. Adding or removing a machine
2. New production ratio
3. Adding new product
4. Changing a resource capacity
5. Resource faults and raw-material processing in a faulty resource
6. The processing routes of the system are changed
7. System contain uncontrollable transitions

When a system has these problems, a system needs to be reconfigurable. Then the deadlock-free system can have deadlocks. Therefore, the proposed robust deadlock control policy needs to be extended to improve efficiency for rapid and valid reconfiguration of Petri net-based supervisory controllers for reconfigurable manufacturing systems.

Author Contributions: Conceptualization, H.K. and A.A.-A.; software, H.K. and R.D.; resources, H.K., A.A.-A. and R.D.; formal analysis, H.K. and A.A.-A.; investigation, H.K., A.A.-A. and Z.L.; validation, H.K., A.A.-A., Z.L. and R.D.; writing—original draft preparation, H.K., A.A.-A. and Z.L.; writing—review and editing, H.K., A.A.-A., Z.L. and R.D.; visualization, H.K., A.A.-A. and Z.L.; supervision, A.A.-A., and Z.L. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, Z.; Zhou, M.; Wu, N. A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2008**, *38*, 173–188.
2. Li, Z.; Wu, N.; Zhou, M. Deadlock control of automated manufacturing systems based on petri nets—A literature review. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 437–462.
3. Abdulaziz, M.; Nasr, E.A.; Al-Ahmari, A.; Kaid, H.; Li, Z. Evaluation of deadlock control designs in automated manufacturing systems. In Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management, Dubai, UAE, 3–5 March 2015; IEEE: Piscataway, NJ, UAE, 2015.
4. Chen, Y.; Li, Z.; Barkaoui, K.; Giua, A. On the enforcement of a class of nonlinear constraints on petri nets. *Automatica* **2015**, *55*, 116–124. [CrossRef]
5. Nasr, E.A.; El-Tamimi, A.M.; Al-Ahmari, A.; Kaid, H. Comparison and evaluation of deadlock prevention methods for different size automated manufacturing systems. *Math. Probl. Eng.* **2015**, *501*, 1–19. [CrossRef]
6. Kaid, H.; Al-Ahmari, A.; El-Tamimi, A.M.; Abouel Nasr, E.; Li, Z. Design and implementation of deadlock control for automated manufacturing systems. *S. Afr. J. Ind. Eng.* **2019**, *30*, 1–23. [CrossRef]
7. Chen, Y.; Li, Z.; Khalgui, M.; Mosbahi, O. Design of a maximally permissive liveness-enforcing petri net supervisor for flexible manufacturing systems. *IEEE Trans. Autom. Sci. Eng.* **2011**, *8*, 374–393. [CrossRef]

8. Wysk, R.A.; Yang, N.-S.; Joshi, S. Detection of deadlocks in flexible manufacturing cells. *IEEE Trans. Robot. Autom.* **1991**, *7*, 853–859. [CrossRef]

9. Ghaffari, A.; Rezg, N.; Xie, X. Design of a live and maximally permissive petri net controller using the theory of regions. *IEEE Trans. Robot. Autom.* **2003**, *19*, 137–141. [CrossRef]

10. Uzam, M.; Zhou, M. Iterative synthesis of petri net based deadlock prevention policy for flexible manufacturing systems. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, Hague, The Netherlands, 10–13 October 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 4260–4265.

11. Uzam, M. The use of the petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems. *Int. J. Adv. Manuf. Technol.* **2004**, *23*, 204–219. [CrossRef]

12. Chao, D.Y. Direct minimal empty siphon computation using mip. *Int. J. Adv. Manuf. Technol.* **2009**, *45*, 397–405. [CrossRef]

13. Chao, D.Y. Improvement of suboptimal siphon-and fbm-based control model of a well-known. *IEEE Trans. Autom. Sci. Eng.* **2011**, *8*, 404–411. [CrossRef]

14. Uzam, M. An optimal deadlock prevention policy for flexible manufacturing systems using petri net models with resources and the theory of regions. *Int. J. Adv. Manuf. Technol.* **2002**, *19*, 192–208. [CrossRef]

15. Chao, D.Y. Fewer monitors and more efficient controllability for deadlock control in s3pgr2 (systems of simple sequential processes with general resource requirements). *Comput. J.* **2010**, *53*, 1783–1798. [CrossRef]

16. Li, Z.; Zhou, M. Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2004**, *34*, 38–51. [CrossRef]

17. Pan, Y.-L.; Tseng, C.-Y.; Row, T.-C. Design of improved optimal and suboptimal deadlock prevention for flexible manufacturing systems based on place invariant and reachability graph analysis methods. *J. Algorithms Comput. Technol.* **2017**, *11*, 261–270. [CrossRef]

18. Zhao, M.; Uzam, M. A suboptimal deadlock control policy for designing non-blocking supervisors in flexible manufacturing systems. *Inf. Sci.* **2017**, *388*, 135–153. [CrossRef]

19. Cong, X.; Gu, C.; Uzam, M.; Chen, Y.; Al-Ahmari, A.M.; Wu, N.; Zhou, M.; Li, Z. Design of optimal petri net supervisors for flexible manufacturing systems via weighted inhibitor arcs. *Asian J. Control* **2018**, *20*, 511–530. [CrossRef]

20. Lautenbach, K. Linear algebraic calculation of deadlocks and traps. In *Concurrency and Nets*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 1987; pp. 315–336.

21. Ezpeleta, J.; Colom, J.M.; Martinez, J. A petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. Robot. Autom.* **1995**, *11*, 173–184. [CrossRef]

22. Davidrajuh, R. *Modeling discrete-event systems with gpensim: An introduction*; Springer International Publishing: Gewerbestrasse, Cham, Switzerland, 2018.

23. Ratzer, A.V.; Wells, L.; Lassen, H.M.; Laursen, M.; Qvortrup, J.F.; Stissing, M.S.; Westergaard, M.; Christensen, S.; Jensen, K. Cpn tools for editing, simulating, and analysing coloured petri nets. In Proceedings of the International Conference on Application and Theory of Petri Nets, Eindhoven, The Netherlands, 23–27 June 2003; Springer: Berlin, Heidelberg, Germany, 2003; pp. 450–462.

24. Li, Z.; Zhou, M. *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*; Springer Science & Business Media: Holborn, London, UK, 2009.

25. Wu, Y.; Xing, K.; Luo, J.; Feng, Y. Robust deadlock control for automated manufacturing systems with an unreliable resource. *Inf. Sci.* **2016**, *346*, 17–28. [CrossRef]

26. Bidgoly, A.J.; Ladani, B.T. Trust modeling and verification using colored petri nets. In Proceedings of the 2011 8th International ISC Conference on Information Security and Cryptology, Mashhad, Iran, 14–15 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–8.

27. Wahab, O.A.; Bentahar, J.; Otrok, H.; Mourad, A. A survey on trust and reputation models for web services: Single, composite, and communities. *Decis. Support Syst.* **2015**, *74*, 121–134. [CrossRef]

28. Wahab, O.A.; Bentahar, J.; Otrok, H.; Mourad, A. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Trans. Serv. Comput.* **2016**, *11*, 184–201. [CrossRef]

29. Wang, N.; Wang, J.; Chen, X. A trust-based formal model for fault detection in wireless sensor networks. *Sensors* **2019**, *19*, 1916. [CrossRef] [PubMed]

30. Piroddi, L.; Cordone, R.; Fumagalli, I. Selective siphon control for deadlock prevention in petri nets. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2008**, *38*, 1337–1348. [CrossRef]

31.　Chen, Y.; Li, Z. Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems. *Automatica* **2011**, *47*, 1028–1034. [CrossRef]

32.　Chen, Y.; Li, Z.; Zhou, M. Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2012**, *42*, 615–629. [CrossRef]