IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 1, 2020, accepted January 15, 2020, date of publication January 21, 2020, date of current version February 5, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2968469*

# Strict Minimal Siphon-Based Colored Petri Net Supervisor Synthesis for Automated Manufacturing Systems With Unreliable Resources

**ABDULRAHMAN AL-AHMARI[1], HUSAM KAID[1], ZHIWU LI[2,3], (Fellow, IEEE), AND REGGIE DAVIDRAJUH[4], (Senior Member, IEEE)**

[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
[3]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[4]Faculty of Science and Technology, University of Stavanger, 4036 Stavanger, Norway

Corresponding authors: Abdulrahman Al-Ahmari (alahmari@ksu.edu.sa) and Husam Kaid (yemenhussam@yahoo.com)

This work was supported by the King Saud University through Researchers Supporting Project Number under Grant RSP-2019/62.

**ABSTRACT** Various deadlock control policies for automated manufacturing systems with reliable and shared resources have been developed, based on Petri nets. In practical applications, a resource may be unreliable. Thus, the deadlock control policies proposed in previous studies are not applicable to such applications. This paper proposes a two-step robust deadlock control strategy for systems with unreliable and shared resources. In the first step, a live (deadlock-free) controlled system that does not consider the failure of resources is derived by using strict minimal siphon control. The second step deals with deadlock control issues caused by the failures of the resources. Considering all resource failures, a common recovery subnet based on colored Petri nets is proposed for all resource failures in the Petri net model. The recovery subnet is added to the derived system at the first step to make the system reliable. The proposed method has been tested using an automated manufacturing system deployed at King Saud University.

**INDEX TERMS** Automated manufacturing system, colored Petri nets, deadlock prevention, siphon.

## I. INTRODUCTION

An automated manufacturing system (AMS) is a collection of buffers, machines, robots, fixtures, and automated tools. There are different products types enter AMS at separate points in time; the system has the ability to handle these products according to the specific sequence of processes and resources sharing. The resource sharing causes deadlocks, in which the local or global system is disabled [1]–[4]. Therefore, in order to prevent deadlock in AMS, an effective deadlock control algorithm is needed.

Petri nets are graphical and mathematical tools that are convenient for modeling, analysis, and control of deadlocks in AMSs [5]–[7]. It utilized to represent the characteristics and behaviors of AMS, such as confliction, synchronization, and sequencing. Moreover, Petri nets can be applied to provide the behavioral characteristics such as boundedness and liveness [8]. In order to solve the deadlock issue in AMSs, several approaches based on Petri nets are proposed in the literature. These methods have been classified into three strategies: deadlock detection and recovery, deadlock avoidance, and deadlock prevention [8]. In addition, three criteria were proposed for evaluating and designing a supervisor for AMS control, namely, computational complexity, structural complexity, and behavioral permissiveness [8]. Therefore, deadlock prevention policies are the objectives of many researchers and can provide liveness-enforcing supervisors with the mentioned criteria [8]. The deadlock control techniques available in the literature were developed for AMSs with reliable and unreliable resources.

For an AMS with reliable resources, there are two techniques used to prevent deadlock involving the use of Petri nets: reachability graph analysis [9]–[12] and structural analysis [13]–[16]. The reachability graph analysis requires listing all or part of the reachable markings; hence, it suffers from a state explosion problem. The reachability graph can be classified into two parts: the live zone (LZ) and the deadlock zone (DZ). First-met bad markings (FBMs) are defined and

The associate editor coordinating the review of this manuscript and approving it for publication was Remigiusz Wisniewski.

extracted from the DZ. In this case, deadlocks are eliminated by designing and adding monitors to prevent FBMs from being reached. This process requires iterations to identify all FBMs [17].

Various approaches have been introduced to prohibit deadlock situations, which are the siphon control and theory of region based approaches [3], [10], [14], [17]–[24]. Structural analysis is implemented to structural objects in Petri nets such as resource transition circuits and siphons. The control steps in this technique are simple: any empty minimal siphon needs an added monitor to prevent itself from being non-empty. Nevertheless, the drawbacks of this technique are that the number of control places is linearly dependent on the net size, and the corresponding controlled system is generally suboptimal. Recently, Guo *et al.* [25] developed an iterative approach for synthesizing a supervisor for an S3PR that enforces liveness. First, two types of emptiable siphons were described and two updated formulations of mixed-integer programming were proposed to calculate these siphons. A three-stage iterative deadlock prevention policy is then proposed, which determines the order of siphon control. The research results indicate that the proposed strategy can obtain a supervisor with lower computational complexity, higher behavioral permissiveness, and simpler structure. In addition, the reachability analysis and the exhaustive siphon enumeration are not needed.

Various deadlock control methods and fault detection have been developed for different classes of Petri nets under unreliable resources [26]–[36]. Lawley and Sulistyono [26] investigated the allocation of resources to a manufacturing system with unreliable resources by designing supervisory control policies that allocate buffer space. This system can manufacture all kinds of products without requiring the use of a failed unreliable resource. Hsieh [27] suggested nominal supervisory algorithms to evaluate the suggested controlled assembly/disassembly Petri net (CADPN) for unreliable resource operations. In addition, Hsieh [27] addressed the conditions under which a system can always operate in the case of resource failures.

Wang *et al.* [28] proposed two policies for single-unit resource allocation systems with unreliable resources. The first policy is for robust operation of one unreliable resource, and the second for several unreliable resources. Chew *et al.* [29] ensured robust operations for systems through the development of two-controller supervisors, in which part types can use a central buffer to handle various unreliable resources. In a study of Liu *et al.* [30], resource failures and deadlocks were considered. First, control places were added to prevent deadlocks in the system model based on a policy of divide-and-conquer. Second, for unreliable resources, recovery subnets were added. Finally, normal and inhibitor arcs were inserted between the recovery subnets and monitors.

Yue *et al.* [31] developed a deadlock controller policy for an AMS class with multiple unreliable resources using Banker's modified algorithm and a set of resource capacity constraints. Another study by Yue *et al.* [32] presented a policy of robust supervisory control in order to prevent the stopping and deadlock of multiple unreliable workstations in the AMS class. Wang *et al.* [33] proposed a deadlock controller for an AMS with resource failures. The supervisor comprises three controllers that ensure that AMS operations with an unreliable resource can be strongly deadlock-free while meeting the required properties.

In the work of Feng *et al.* [34], deadlocks were described and established in the context of maximal perfect resource transition circuits, where the authors exposed a novel concept of powerful transition cover in order to develop a deadlock controller with a simple structure. Liu *et al.* [37] proposed two robust block control policies based on a reachability graph with high behavioral permissiveness, strong robustness, or simple design for an AMS. The technique of reachability graph partition was used to analyze robust legal and prohibited markings. Through this, an issue of deadlock control was converted into a problem of preventing prohibited markings. For a convex legal space of reachability, a robust control method was provided. The robust legal space of reachability, however, may be nonconvex.

Li *et al.* [37] developed a two-step deadlock control policy and a robust legal marking. In the first step, control places are designed based on an elementary siphon policy developed in [38], which ensures that the system model is deadlock free if there is no failure in resource. The second step addresses failure-induced deadlock control problems. The resource failures are modeled by recovery subnets and recoveries are added into the first-step-derived system, resulting in an unreliable controlled system. In previous deadlock control methods, control places are developed to prohibit deadlocks; recovery subnets are proposed to model resource failures and recoveries and applied for each unreliable resource, resulting in high structural complexity of the initial model. However, there is a need to propose a supervisor to address all the unreliable resources in a system. This supervisor does not require to introduce inhibitor arcs or enumerate of reachability graphs and leads to low computational overheads.

Therefore, the aim of this article is to design a two-step deadlock control policy for systems with unreliable resources. In the first step, a controlled system without considering the failures of resources is derived by using strict minimal siphons (SMSs) proposed by Ezpeleta *et al.* [39] to build a controlled Petri net model. The SMSs based policy can obtain a maximal permissive liveness-enforcing supervisor for larger sized systems [5] compared with previous deadlock prevention methods [19], [40]. In the second step, one common recovery subnet based on colored Petri nets is designed to model all resource failures, and recovery is added to the derived system by the first step in order to make the system reliable.

The paper is organized as follows. Section II describes basic concepts of Petri nets and a deadlock prevention policy based on strict minimal siphon. The robust control of

unreliable resources based on colored Petri nets and computational complexity of the proposed policies are presented in Section III. The General Petri Net Simulator (GPenSIM) code and validation for the developed method is presented in Section V. A real-world AMS case study is given in Section VI, followed by the conclusions and future research presented in Section IV.

## II. PRELIMINARIES

This section presents the basics of Petri nets, strict minimal siphons, a deadlock prevention policy based on SMSs, and a GPenSIM tool.

### A. 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 graphically named by nodes, where $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$; places are depicted by circles and transitions are depicted by bars. $F \subseteq (P \times T) \cup (T \times P)$ is said to be a set of directed arcs of $N$ that join the places to transitions or transitions to places. $W : (P \times T) \cup (T \times P) \rightarrow \boldsymbol{IN}$ is a mapping that assigns an arc's weight, where $\boldsymbol{IN} = \{0, 1, 2, ...\}$.

$N = (P, T, F, W)$ is called an ordinary net if $\forall(p, t) \in F$, $W(p, t) = 1$. Let $x, z \in P \cup T$ be nodes in $N = (P, T, F, W)$, $N$ is said to be a weighted net if there is an arc between $x$ and $z$ such that $W(x, z) > 1$. Assume that $a$ and $b$ are nodes in $N = (P, T, F, W)$, i.e., $a, b \in P \cup T$. Then, ${}^\bullet a = \{b \in P \cup T | (b, a) \in F\}$ is called the input (preset) of node $a$, and $a^\bullet = \{b \in P \cup T | (a, b) \in F\}$ is called the output (postset) of node $a$.

A marking $M$ of $N$ is a mapping $M : P \rightarrow \boldsymbol{IN}$ and represents the status of the system. $(N, M_o)$ is a marked Petri net, expressed as $(N, M_o)$ $(P, T, F, W, M_o)$, where $M_o$ is the initial marking of $N$, $M_o : P \rightarrow \boldsymbol{IN}$. Suppose that $M(p)$ is the number of tokens in place $p$, a transition $t \in T$ is enabled at any marking $M$ if for all $p \in {}^\bullet t$, $M(p) \geq W(p, t)$, which is expressed as $M[t\rangle$. When the enable transition $t$ fires, it takes $W(p, t)$ token(s) from each place $p \in {}^\bullet t$, and deposits $W(t, p)$ token(s) in each place $p \in t^\bullet$. Thus, it leads the system to reach a new marking $M'$, expressed as $M[t\rangle M'$ in short, the new reachable marking can be computed as $M'(p) = M(p) - W(p, t) + W(t, p)$. We call $N = (P, T, F, W, M_o)$ 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 that represents the Petri net $N$ structure, where $[N]$ is an integer matrix that consists of $|T|$ columns and $|P|$ rows and computed from the input and output matrices, expressed as $[N](p, t) = W(t, p) - W(p, t)$.

$R(N, M)$ is a set of markings that are reachable from $M$ in a Petri net model $(N, M)$. Let $(N, M_o)$ be a marked Petri net with $N = (P, T, F, W, M_o)$. A transition $t \in T$ is live (deadlock free) if for all $M \in R(N, M)$, there exists $M' \in R(N, M)$ such that firing sequence $M'[t\rangle$ 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 called reachable from $M$ if there exist a finite transition sequence $\delta = t_1, t_2, t_3, \ldots, t_n$ that can be fired, and markings $M_1, M_2, M_3, ...,$ *and* $M_{n-1}$ such that $M[t_1\rangle M1[t_2\rangle M_2[t_3\rangle M_2 \ldots M_{n-1}[t_n\rangle M'$, which is represented as $M[\delta\rangle M'$, satisfy the state equation $M' = M + [N]\vec{\delta}$, where $\vec{\delta}: T \rightarrow \boldsymbol{IN}$ maps $t$ in $T$ to the number of occurrences of $t$ in $\delta$ and is named a Parikh vector.

Let $(N, M_o)$ be a marked Petri net with $N = (P, T, F, W, M_o)$. A net $N$ with initial marking $M_o$ is called $k$-bounded if for all $M \in R(N, M_0)$, for all $p \in P$, $M(p) \leq k (k \in \{1, 2, 3, \ldots\})$. A net $N$ is safe if all of its places are safe, i.e., in each place $p$, the number of tokens does not exceed one. A net $N$ is $k$-safe if it is $k$-bounded.

A place vector (P-vector) $I: P \rightarrow \boldsymbol{Z}$ indexed by $P$ is called a place invariant (P-invariant) if $I^T . [N] = \boldsymbol{0}^T$ and $I \neq \boldsymbol{0}$, a transition vector (T-vector) $J : T \rightarrow \boldsymbol{Z}$ indexed by $T$ is called a transition invariant (T-invariant) if $[N] . J = \boldsymbol{0}$ and $J \neq \boldsymbol{0}$, where $\boldsymbol{Z} = \{\ldots, -2, -1, 0, 1, 2, ...\}$. A place invariant $I$ is said to be a place semiflow or P-semiflow, if each element of $I$ is nonnegative. Let us assume $I$ is a P-invariant of a $N = (P, T, F, W, M_o)$ and $M$ is a marking reachable from the initial marking $M_o$. Then, $I^T M = I^T M_o$. The support of P-invariant $I$ is expressed as Let $||I|| = \{p \, | I(p) \neq 0\}$. The support of T-invariant $J$ is expressed as Let $||J|| = \{t \, | J(t) \neq 0\}$. The $||I||$ are categorized into three types: (1) The positive support of P-invariant $I$ and expressed as $||I||^+ = \{p \, | I(p) > 0\}$. (2) The negative support of P-invariant $I$ and expressed as $||I||^- = \{p \, | I(p) < 0\}$. (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 expressed as a nine-tuple [16] $CPN = (P, T, F, SC, C_f, N_f, A_f, G_f, I_f)$, where $P$, $T$, and $F$ are defined above. $SC$ is a set of colors that comprises colors $c_i$ and the operations on the $c_i$. $C_f$ is the color function that traces $p_i$ into colors $c_i$, $p_i \in P$ and $c_i \in SC$. $N_f$ is the node function that traces $F$ into $(P \times T) \cup (T \times P)$. $A_f$ is the arc function that traces each flow (arc) $f \in F$ into the term $e$. $G_f$ is the guard function that traces each transition $t_i \in T$ to a guard expression $g$ that has a Boolean value. $I_f$ is the initialization function that traces each place $p_i \in P$ into an initialization expression.

### B. STRICT MINIMAL SIPHONS

*Definition 1 [38]:* A simple sequential process ($S^2P$) is a Petri net model with $N = (\{p^0\} \cup P_A, T, F)$, satisfying: (1) $N$ is a strongly connected state machine and (2) each circuit $N$ contains place $p^0$, where $p^0$ is a process idle place and $P_A$ is a set of operation places, $P_A \neq \emptyset$.

*Definition 2 [38]:* A simple sequential process with resources ($S^2PR$) is a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$, satisfying:

1. The subnet created by $Y = P_A \cup \{p^0\} \cup T$ is an $S^2P$.
2. $P_R \neq \emptyset$ an $(P_A \cup \{p^0\}) \cap P_R = \emptyset$, where $P_R$ is called a set of resource places.
3. $\forall p \in P_A$, $\forall t \in {}^\bullet p$, $\forall t' \in p^\bullet$, $\exists r_p \in P_R$, ${}^\bullet t \cap P_R = t'^\bullet \cap P_R = \{r_p\}$.

4. $\forall r \in P_R, {}^{\bullet\bullet}r \cap P_A = r^{\bullet\bullet} \cap P_A \neq \emptyset$ and ${}^{\bullet}r \cap r^{\bullet} \neq \emptyset$.
5. ${}^{\bullet\bullet}(p^0) \cap P_R = (p^0)^{\bullet\bullet} \cap P_R \neq \emptyset$.

*Definition 3 [38]:* A simple sequential process with resources ($S^2$PR) with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$, and $M_o$ is called an initial marking of net $N$. An $S^2$PR is called acceptably marked, satisfying: (1)$M_o(p^0) \geq 1$, (2) $M_o(p) = 0$, $\forall p \in P_A$, and (3) $M_o(r) \geq 1$, $\forall r \in P_R$.

*Definition 4 [38]:* Let $S^2$PR a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$ and called $S^3$PR for abbreviation, is repetitively defined as follows:

1. An $S^2$PR is as well an $S^3$PR
1. Let $N_1$ and $N_2$ be two $S^3$PRs, where $N_1 = (\{p_1^0\} \cup PA_1 \cup PR_1, T_1, F_1)$ and $N_2 = (\{p_2^0\} \cup PA_2 \cup PR_2, T_2, F_2)$, such tha $(\{p_1^0\} \cup PA_1) \cap (\{p_2^0\} \cup PA_2) = \emptyset$, $PA_1 \cap PA_2 \neq P_C$, $PR_1 \cap PR_2 = P_C$ and $T_1 \cap T_2 \neq \emptyset$. Then, the net $N = (\{p^0\} \cup P_A \cup P_R, T, F)$ is an $S^3$PR resulting from the integration of $N1$ and $N2$ by the set of common $P_C$ and expressed as: (1) $p^0 = \{p_1^0\} \cup \{p_2^0\}$, $P_A = PA_1 \cup PA_2$, $P_R = PR_1 \cup PR_2$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$.

The integration of $n$ $S^2$PR $N_1$-$N_n$via $P_C$, is expressed by $\otimes_{i=1}^{n} N_i$. $\overline{N_i}$ is used to indicate the $S^2$P from which the $S^2$PR $N_i$ is built.

*Definition 5 [38]:* Let $N$ be an $S^3$PR a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$. $M_o$ is an initial marking of $N$. $(N, M_o)$ is called acceptably marked, satisfying: (1)$(N, M_o)$ is an acceptably marked $S^2$PR. (2) $N = N_1 \circ N_2$, where $(N_i, M_{oi})$ is called an acceptably marked $S^3$PR and

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

*Definition 6 [38]:* Let $N$ be an $S^3$PR a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$. A non-empty set $S \subseteq P$ is named a siphon in $N$ if ${}^{\bullet}S \subseteq S^{\bullet}$. If a siphon contains no other siphons, it is considered a minimal siphon.

*Definition 7 [38]:* Let $N$ be an $S^3$PR a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$. $S$ is named a minimal siphon in $N$. A minimal siphon $S$ is named a strict minimal siphon if $S^{\bullet} \subsetneq {}^{\bullet}S$. Let $\Pi = \{S_1, S_2, S_3, \ldots, S_k\}$ be a set of strict minimal siphons of $N$. We have $S = S_A \cup S_R$, $S_R = S \cap P_R$, and $S_A = S \setminus S_R$, where $S_A$ and $S_R$ are sets of operations and resources places, respectively.

*Definition 8 [38]:* Let $N$ be an $S^3$PR a Petri net model with $N = (\{p^0\} \cup P_A \cup P_R, T, F)$, $r \in P_R$ be a reliable resource place in $N$. The operation places that use $r$ are recognized as the set of holders of $r$, expressed as $H(r) = \{p \setminus p \in P_A, p \in {}^{\bullet\bullet} r \cap P_A \neq \emptyset\}$. $[S]$ is called the complementary set or stealing places of $S$ if $[S] = (\cup_{r \in SR} H(r)) \setminus S_A$, where stealing places are operation places that require resources places of siphon $S$, but are not in the siphon $S$.

Consider the $S^3$PR Petri net model displayed in Figure 1. The Petri net model includes six places and four transitions. The places can be expressed as the following sets: $P^0 = \{p_1\}$, $P_R = \{p_5, p_6\}$, and $P_A = \{p_2, p_3, p_4\}$. Note that the transitions in $(P^0)^{\bullet}$ are called source transitions that represent the entry of raw materials when a manufacturing system is modeled with an $S^3$PR. The transitions in ${}^{\bullet}(P^0)$ are called sink transitions that represent the exist of finished materials when a manufacturing system is modeled with an $S^3$PR. The Petri net model has four minimal siphons, one of which is $S = \{p_4, p_5, p_6\}$ that is strict minimal $S_3 = \{P_3P_7P_9P_{10}$. According to Definitions 7 and 8, we have

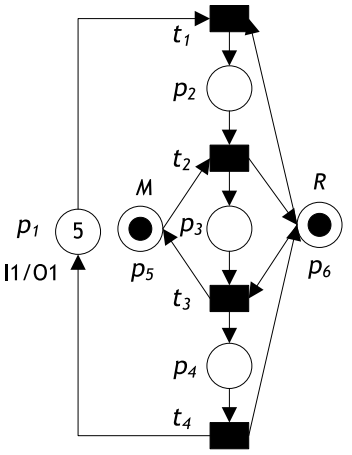


**FIGURE 1.** $S^3$PR Petri net model.

1) $S_A = \{p_4\}$
2) $S_R = \{p_5, p_6\}$
3) $H(p_5) = \{p_3\}$, $H(p_6) = \{p_2, p_4\}$.
4) $[S] = \{p_2, p_3\}$.

### C. DEADLOCK PREVENTION POLICY BASED ON STRICT MINIMAL SIPHONS

This section introduces a deadlock-prevention approach based on SMSs to build a controlled Petri net model. This approach is derived from Ezpeleta *et al.* [39].

*Definition 9 [38]:* Let $(N_1, M_1)$ and $(N_2, M_2)$ be are two Petri nets with $N_i = (P_i, T_i, F_i, W_i)$, where $i = 1, 2$ . $(N, M)$ with $N = (P, T, F, W)$ is named a synchronous net resulting from the integration of $(N_1, M_1)$ and $(N_2, M_2)$, defined as $(N_1, M_1) \parallel (N_2, M_2)$, satisfying: (1) $P = P_1 \cup P_2$ and $P_1 \cap P_2 = \emptyset$.(2) $T = T_1 \cup T_2$, (3) $F = F_1 \cup F_2$, (4) $W(e) = W_i(e)$, where $e \in F_i$, $i = 1, 2$, and (5) $M(p) = M_i(p)$, $p \in P_i$, $i = 1, 2$.

*Definition 10 [38]:* Let $(N, M_o)$ be an $S^3$PR with $N = (\{p^0\} \cup P_A \cup P_R, T, F, M_o)$. The deadlock controller for $(N, M_o)$ proposed in Ezpeleta *et al.* [39] is defined as $(V, M_{Vo}) = (P_V, T_V, F_V, M_{Vo})$, where (1)$P_V = \{V_S \setminus S \in \Pi\}$ is set of control places.(2)$T_V = \{t \setminus t \in {}^{\bullet}V_S \cup V_S^{\bullet}\}$. (3)$F_V \subseteq (P_V \times T_V) \cup (T_V \times P_V)$ is named a flow relation of $N$, denoted by an arc with an arrow from control places to transitions or transitions to control places. (4) For all $V_S \in P_V$, $M_{Vo}(V_S) = M_{Vo}(S) - 1$, where $M_{Vo}(V_S)$ is an initial marking of a control place.

$(N_V, M_{Vo})$ is named a controlled Petri net model resulting from the composition of $(V, M_{Vo})$ and $(N, M_o)$, defined as $(V, M_{Vo}) \parallel (N, M_o)$. A monitor or control place is added to each SMS to fulfill the liveness of a Petri net and to make sure that no SMSs can ever be emptied. The proposed policy is easy and ensures success. Nevertheless, it leads to a more

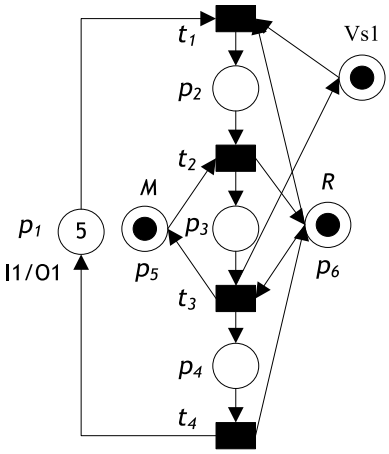


**FIGURE 2.** Controlled $S^3PR$ Petri net model.

Policy 1 (Deadlock-Prevention Algorithm Based on SMSs)

**Input:** An $S^3PR$ $(N, M_o)$.
**Output:** A controlled net $(N_V, M_{Vo})$
**Step 1:** Compute the set $\Pi$ for $N$.
**Step 2: for** each $S \in \Pi$ **do**

- Add a control place $V_S$. / By using Definition 10 . */
- Add $Vs$ output arcs that connected to the source transitions, which are led to the sink transitions of $S$, and all arc weights are unitary. / By using Definition 10 . */
- Add $Vs$ input arcs that connected from the stealing places of $S$, and 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**

complex Petri-net-controlled system than the original Petri net model because the number of inserted control places is equal to that of the SMSs in the target Petri net model, and the inserted arcs are greater in number than the added control places.

Based on the strict minimal siphon concept, the deadlock prevention algorithm developed by Ezpeleta *et al.* [39] is shown as follows:

Reconsider Figure 1. One siphon is $S = \{p_4, p_5, p_6\}$. By Definition 8, its complementary set is $[S] = \{p_2, p_3\}$. Based on Definition 10 and Policy 1, monitor $V_{S1}$ needs to be added for $S$, with ${}^\bullet V_{S1} = \{t_3\}$, $V_{S1}^\bullet = \{t_1\}$, and $M_{Vo}(V_{S1}) = 1$. Figure 2 displays the controlled Petri net model after adding the monitor.

### D. GPenSIM TOOL

GPenSIM Petri net simulator is used in this paper as conceived by Davidrajuh [41]. It is possible to run GPenSIM on MATLAB platform. GPenSIM was developed to model, simulate, control, and analyze discrete event systems. GPenSIM can integrate toolboxes in MATLAB with Petri net models (e.g., "Fuzzy logic", optimization tools, and "Control systems"). In GPenSIM the functions $C_f$, $N_f$, $A_f$, $G_f$, and $I_f$ are integrated together and are coded in the preprocessor files. In addition, logical expressions and constraints can only be processed by transitions. GPenSIM uses three files to model, simulate, analyze, and control the Petri net models, which are [16]:

1. Petri net definition file (PDF) that used to describe the structure of a Petri net model (places, transitions, and arcs).
2. Main simulation file (MSF) that used to simulate a Petri net model that defied in PDF file
3. Pre- and postprocessor files that used to inspect and control the conditions of firing for a certain transition are met, and execute post-firing activities if needed after a certain transition has been fired during the simulation.

In Petri net model, inside any place $p$, all tokens are homogeneous, does not matter which token was first or last to arrive at the place, and which transition a token is deposited at the place. Nevertheless, in GPenSIM each token can become unique with identification number (***tokID***). In addition, it can assign some identifiers ("colors") to each token. The colors of the output tokens can be added, changed, or deleted in the preprocessor file by only transitions $T$. When a transition $t$ fires, all colored tokens from the input places transfer to the output places. However, colors are inherited and can be avoided by overriding. Moreover, transition can select specific color-based input tokens [16].

Finally, in GPenSIM each token has the following structure: (1) ***tokID***: a single token identifier (integer value); (2) ***t_color*** (text string set) is a color setting. Several GPenSIM functions are used to manipulate the colors. One of the functions used in this paper is ***tokenEXColor*** that can be described 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. Input arguments are ***Place***( from which place the tokens are to be chosen), ***nr_tokens_wanted*** (the required number of tokens that have a specified color), and **t_color** (a colors set). Output values are ***set_of_tokID*** (a set of ***tokIDs*** that satisfy the color requirements) and ***nr_token_av*** (the number of ***tokIDs*** existing in ***set_of_tokID***) *[16].*

## III. ROBUST CONTROL FOR UNRELIABLE RESOURCES BASED ON COLORED PETRI NETS

This section describes a novel robust two-step control policy. In the first step, the system's resources are generally assumed to be reliable. Several monitors are added after applying a strict minimal siphon-based control strategy to such a system. In the second step, by considering that resources may fail, a common recovery subnet is added to model all resource failures in a system. As a result, a robust controlled system

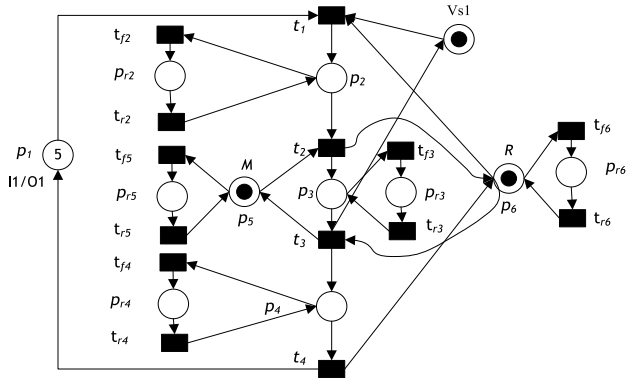


**FIGURE 3.** Unreliable Petri net model by definition 11.

is developed. Section II presents the method used in the first step. This section concentrates on the relevance between resource failures and the controlled system in the first step.

### A. COMMON RECOVERY SUBNET BASED ON COLORED PETRI NETS

*Definition 11 [30]:* Let $r_u \in P_R$ be an unreliable resource. A recovery subnet of $r_u$ is a PN $N_{ri}$= ($\{p_i, p_{ri}\}$, $\{t_{fi}, t_{ri}\}$, $F_{ri}$), where $F_{ri}$= {( $p_i$, $t_{fi}$), ($t_{fi}$, $p_{ri}$), ($p_{ri}$, $t_{ri}$), ($t_{ri}$, $p_i$)}, and an unreliable resource may fail when it is idle $r_u$ or in a busy state (its holders), $p_i \in \{r_u\} \cup H(r_u)$. ($N_{ri}$, $M_{rio}$) is called a marked recovery subnet, where $M_{rio}(p_i) \geq 0$ and $M_{rio}(p_{ri}) = 0$.

In Definition 11, $p_{ri}$ is called the recovery place of all $p_i$. Transitions $t_{fi}$ and $t_{ri}$ indicate that an unreliable resource $r_u$ fails in $p_i$ and recovers through $p_{ri}$, respectively. By Definition 11, for Figure 2, we can add a recovery subnet for each unreliable resource $r_u$ in each case an idle or busy state results in an unreliable Petri net model as depicted in Figure 3.

The main weakness of the Definition 11 in [30] is that many recovery subnets are added for all unreliable resources, which leads to high structural complexity of the initial model. However, an efficient approach is developed in this section to minimize the computational overhead.

*Definition 12:* Let $r_u \in P_R$ be an unreliable resource. A colored common recovery subnet of $r_u$ is a PN $N_{cri}$= ($\{p_i, p_{combined}\}$, $\{t_{fi}, t_{ri}\}$, $F_{cri}$, $C_{cri}$), where $F_{cri}$= {($p_i$, $t_{fi}$), ( $t_{fi}$, $p_{combined}$), ($p_{combined}$, $t_{ri}$), ($t_{ri}$, $p_i$)}, and an unreliable resource may fail when it is idle $r_u$ or in a busy state (its holders). Thus, we define $P_{RH}$= $\{r_u\} \cup H(r_u)$ as a set of places, where $H(r_u)$ is a set of holders of $r_u$, indicated by $H(r_u) = \{p \backslash p \in P_A, p \in {}^{\bullet\bullet} r_u \cap P_A \neq \emptyset\}$, $p_i \in P_{RH}$. $C_{cri}$ is the color that maps $p_i \in P_{RH}$ into colors $C_{cri} \in SC$. ($N_{cri}$, $M_{crio}$) is called a colored common marked recovery subnet, where $M_{crio}(p_i) \geq 0$ and $M_{crio}(p_{combined}) = 0$.

In Definition *12*, $p_{combined}$ is called the recovery place of all $p_i$. Transitions $t_{fi}$ and $t_{ri}$ indicate that an unreliable resource $r_u$ fails in $p_i$ and recovers through $p_{combined}$, respectively. If an unreliable resource fails in $p_i$, then the token in $p_i$ flows into $p_{combined}$ by firing $t_{fi}$. When transition $t_{fi}$ fires, it adds a color $C_{cri}$ to the tokens from $p_i$ and deposits them into the common place $p_{combined}$. After the failed resource is repaired, the colored token in $p_{combined}$ flows into $p_i$ by firing $t_{ri}$. When transition $t_{ri}$ fires, it selects only the tokens with color $C_{cri}$ from $p_{combined}$ and deposits them into $p_i$, indicating that a resource recovery is finished. Note that by default, colors are inherited: when a transition $t_{ri}$ fires, it collects all colors from the consumed ($p_{combined}$) tokens and passes them to the deposited ($p_i$) tokens. However, color inheritance can be prevented by overriding.

*Definition 13:* Let ($N_V$, $M_{Vo}$) be a controlled $S^3$PR, and $P_{Ru}$ be the set of unreliable resources. For all $r_u \in P_{Ru}$, adding one common recovery subnet for all $p_i \in P_{RH}$ results in a colored controlled unreliable Petri net defined as $(N_C, M_{Co}) = (N_V, M_{Vo}) \parallel (N_{cri}, M_{crio})$ that is the composition of ($N_V$, $M_{Vo}$) and ($N_{cri}$, $M_{crio}$).

*Definition 14:* Let $N_C = (P^0 \cup P_A \cup P_R \cup P_V \cup p_{combined}, T \cup T_F \cup T_R, F \cup F_R, C_R, M_{Co})$ be a colored controlled unreliable marked $S^3$PR, and $R(N_C, M_{Co})$ be its reachable graph, where $T$, $T_F$, $T_R$, $C_R$, and $M_{Co}$ are the operation transitions, failure transitions, recovery transitions, failure colors in failure transitions, and the initial marking of $N_C$, respectively. $F_R \subseteq (T_R \times P_{RH}) \cup (P_{RH} \times T_F)$.

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

*Proof:* There is a need to prove that all transitions in $T$, $T_F$, and $T_R$ in ($N_C$, $M_{Co}$) are live. It does not matter how a controlled system develops, and there are no strict minimal siphons being emptied. In addition, there is no new strict minimal siphon being created, which indicates that the net ($N_C$, $M_{Co}$) is live when there is no failure in an unreliable resource $r_u \in P_{Ru}$ because all $t_1 \in T$ are live. The net ($N_C$, $M_{Co}$) is live when at least one unreliable resource fails but at least one part type can still be processed by the colored controlled unreliable Petri net ($N_C$, $M_{Co}$).

Moreover, if a failure in the unreliable resource $r_u \in P_{Ru}$ occurs, then the failed resource $r_u$ will be repaired successfully, the system may be returned to operate without causing a deadlock, and the colored controlled unreliable Petri net ($N_C$, $M_{Co}$) will remain live for all $t_2 \in T_F$ if for all $p_i \in {}^{\bullet} t_2$, $M_C(p_i) > 0$. Then, $t_{fi}$ can fire in any case because it is uncontrollable, leading to $M_C(p_{combined}) > 0$ for all $t_3 \in T_R$. If $M_C(p_{combined}) > 0$, then $t_3$ can fire. It can therefore be said that the colored controlled unreliable Petri net ($N_C$, $M_{Co}$) is live.

Based on strict minimal siphons, unreliable resources, and colored Petri nets, the developed policy is shown as follows:

Considering an unreliable marked $S^3$PR net shown in Figure 2, assume that the index set that may be used is NA = $\{i | p_i \in P_{RH}\}$. $T_F = \cup_{i \in NA}\{t_{fi}\}$, $T_R = \cup_{i \in NA}\{t_{ri}\}$, and $C_F = \cup_{i \in NA}\{C_{cri}\}$, where $t_{fi}$, $t_{ri}$, and $C_{cri}$ are defined by using Definitions 11 and 12. In the net shown in Figure 2, we have $P_{Ru}$= $\{p_5, p_6\}$, $H(p_5) = \{p_3\}$, and $H(p_6) = \{p_2, p_4\}$. Adding a common recovery subnet for $p_5$ and $p_6$ by Definition 12 results in an unreliable $S^3$PR net model, as depicted in Figure 4. NA = {2, 3, 4, 5, 6},

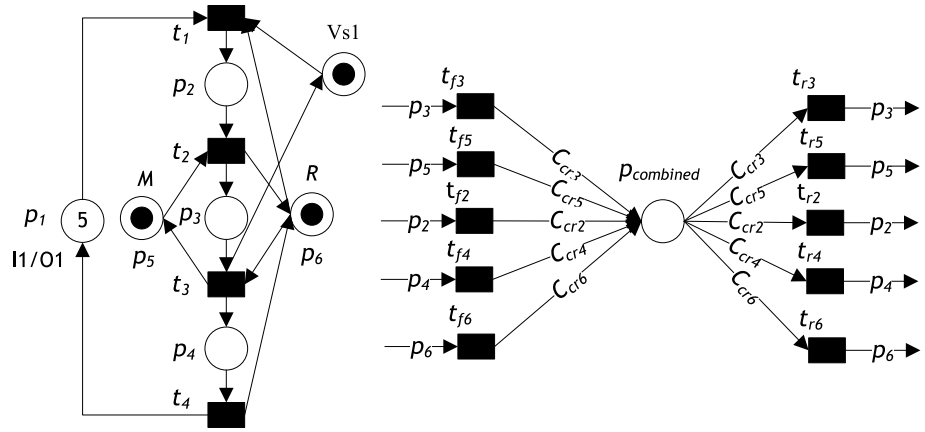


**FIGURE 4.** Colored controlled unreliable Petri net model by Policy 2.

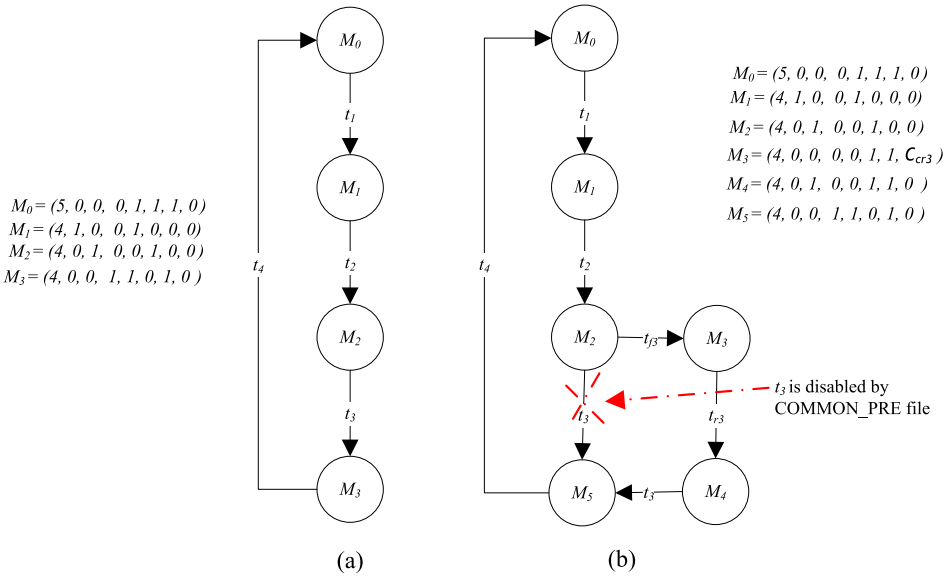


**FIGURE 5.** (a) Reachability graph of case 1 and (b) Reachability graph of case 2.

$T_F = \{t_{f2}, t_{f3}, t_{f4}, t_{f5}, t_{f6}\}$, $T_R = \{t_{r2}, t_{r3}, t_{r4}, t_{r5}, t_{r6}\}$, and $C_R = \{C_{cr2}, C_{cr3}, C_{cr4}, C_{cr5}, C_{cr6}\}$.

If resource $p_5$ fails in the busy state $p_3$, i.e., $t_{f3}$ fires, then it adds a color $C_{cr3}$ to the token from $p_3$ and deposits it into the common place $p_{combined}$. If resource $p_5$ fails in idle state $p_5$, i.e., $t_{f5}$ fires, then it adds a color $C_{cr5}$ to the token from $p_5$ and deposits it into the common place $p_{combined}$. After the failed resource $p_5$ is repaired, the colored token in $p_{combined}$ flows into $p_3$ or $p_5$ by firing $t_{r3}$ or $t_{r5}$. When the transition $t_{r3}$ or $t_{r5}$ fires, selects only the tokens with color $C_{cr3}$ or $C_{cr5}$ from $p_{combined}$ and deposits them into $p_3$ or $p_5$, indicating that a resource $p_5$ recovery is finished.

If resource $p_6$ fails in busy state $p_2$ or $p_4$, i.e., $t_{f2}$ or $t_{f4}$ fires, it adds a color $C_{cr2}$ or $C_{cr4}$ to the token from $p_2$ or $p_4$ and deposits them into the common place $p_{combined}$. If resource $p_6$ fails in idle state $p_6$, i.e., $t_{f6}$ fires, then it adds a color $C_{cr6}$ to the token from $p_6$ and deposits it into the common place $p_{combined}$. After the failed resource $p_6$ is repaired, the colored token in $p_{combined}$ flows into $p_2$, $p_4$, or $p_6$ by firing $t_{r2}$, $t_{r4}$, or $t_{r6}$ when the transition $t_{r2}$, $t_{r4}$, or $t_{r6}$ fires, and selects only the tokens with colors $C_{cr2}$, $C_{cr4}$, or $C_{cr6}$ from $p_{combined}$ and deposits them into $p_2$, $p_4$, or $p_6$, indicating that the recovery of resource $p_6$ is finished.

The subnet designed in the second step cannot cause new deadlocks. To prove that it needs only to examine the liveness of a net system. Consider the colored controlled unreliable Petri net model shown in Figure 4. There are two cases in a model, which are:

1) If there is no failure in any unreliable resource $r_u \in P_{Ru}$, thus, all $t \in T$ are live (deadlock-free), where $T = \{t_1, t_2, t_3, t_4\}$ denoted as operation transitions. In addition, all failure transitions $T_F = \{t_{f2}, t_{f3}, t_{f4}, t_{f5}, t_{f6}\}$ and recovery transitions $T_R = \{t_{r2}, t_{r3}, t_{r4}, t_{r5}, t_{r6}\}$ are disabled by developed preprocessor (COMMON_PRE file in GePenSim tool). The COMMON_PRE file will inspect if the conditions of firing for failure transitions

Policy 2 (Robust Control for Unreliable Resources Based on Colored Petri Net Algorithm)

**Input:** A controlled net $(N_V, M_{Vo})$ by Policy 1

**Output:** A colored controlled unreliable marked Petri net $(N_C, M_{Co})$

**Step 1: for** each $r_u \in P_{Ru}$ **do**

- Add a transition to represent breakdown resource $t_{fi}$. / By using Definition 12 . ∗/
- Define colors $C_{cri}$ for failure transition $t_{fi}$. / By using Definition 12. ∗/
- Add a recovery place $p_{combined}$. / By using Definition 12. ∗/
- Add a recovery transitions to represent that the failed resource is repaired $t_{ri}$. / By using Definition 12. ∗/
- The $p_i$ output arcs are connected to the $t_{fi}$ transition, and all arc weights are unitary. / By using Definition 12. ∗/
- The $p_{combined}$ input arcs are connected from the $t_{fi}$ transitions, and all arc weights are unitary. / By using Definition 12. ∗/
- The $p_{combined}$ output arcs are connected to the $t_{ri}$ transitions, and all arc weights are unitary. / By using Definition 12. ∗/
- The $p_i$ input arcs are connected from the $t_{ri}$ transitions, and all arc weights are unitary. / By using Definition 12. ∗/

**end for**

**Step 2**: Output a colored controlled unreliable marked Petri net $(N_C, M_{Co})$.

**Step 3: End**

are met (inspect time to failure). Figures 5(a) shows the reachability graph of case 1, and the system is live.

2) The colored controlled unreliable Petri net $(N_C, M_{Co})$ is live (deadlock- free) when at least one unreliable resource fails but at least one part type can still be processed by the colored controlled unreliable Petri net $(N_C, M_{Co})$. Assume that if a failure in the unreliable resource $p_5$ occurs at busy state $p_3$, then the failed resource $p_5$ will be repaired successfully, the system may be returned to operate without causing a deadlock, and the net $(N_C, M_{Co})$ will remain live. Figures 5(b) shows the reachability graph of case 2. If resource $p_5$ fails in the busy state $p_3$, i.e., $t_{f3}$ fires, then it adds a color $C_{cr3}$ to the token from $p_3$ and deposits the token into the common place $p_{combined}$. After the failed resource $p_5$ is repaired, the colored token in $p_{combined}$ flows into $p_3$ by firing $t_{r3}$. When the transition $t_{r3}$ fires, it selects only the token with color $C_{cr3}$ from $p_{combined}$ and deposits the token into $p_3$, indicating that a resource $p_5$ recovery is finished. Then, $t_3$ resumes to be fired. Note that the conditions for the enabled failure and recovery transitions to start firing based on mean time to failure and mean time to repair, respectively.

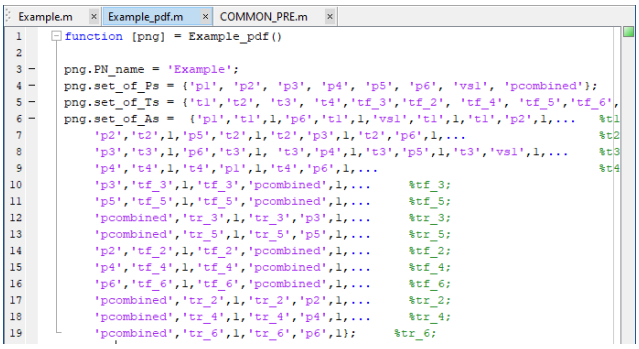


**FIGURE 6.** PDF file of colored controlled unreliable Petri net model in Figure 3.

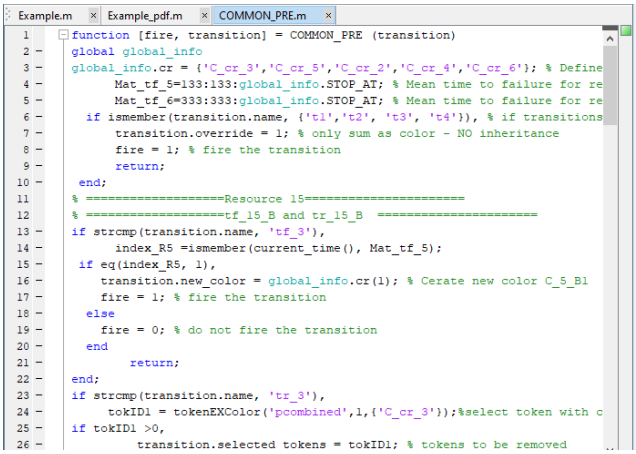


**FIGURE 7.** MSF file of colored controlled unreliable Petri net model in Figure 3.

### B. COMPUTATIONAL COMPLEXITY

Policy 1 computes the control places to S$^3$PR $(N, M_o)$ with $N = (P_A \cup \{p^0\} \cup P_R, T, F, M_o)$. Policy 1 is used to add a control place $V_S$ to each SMS in S$^3$PR $(N, M_o)$ to fulfill the liveness of a Petri net. Obviously, each $V_S$ is associated with an SMS in S$^3$PR $(N, M_o)$. Let $n$ is the number of SMSs i.e., $|\text{SMSs}| = n$. The "FOR loop" loop is executed $n$ times to add $V_S$ for the SMSs in S$^3$PR $(N, M_o)$. Thus, the computational complexity of Policy 1 to add control places to S$^3$PR $(N, M_o)$ is O($n$). In Definition 11 [30], a recovery subnet is computed for each unreliable resource in controlled S$^3$PR $(N_V, M_{Vo})$. Based on the fact that each unreliable resource is associated with the recovery subnet in S$^3$PR $(N_V, M_{Vo})$, each unreliable resource $r_u \in P_{Ru}$ may fail when it is idle $r_u$ or in a busy state $H(r_u)$. Thus, $P_{RH} = \{r_u\} \cup H(r_u)$ as a set of places. Let the number of set of places that need recovery subnets $P_{RH}$ be $y$, i.e., $|P_{RH}| = y$. Moreover, each recovery subnet requires failure transition, recovery place, and recovery transition. Assume that the number of recovery subnet items is *3*. The "FOR loop" is executed $3y$ times to compute the recovery subnets for all unreliable resource in controlled S$^3$PR $(N_V, M_{Vo})$. Thus, the computational complexity is O($3y$). While in the proposed Policy 2, the recovery subnet requires failure transition and recovery transition for each

```
Example.m | Example_pdf.m | COMMON_PRE.m
clear all; clc;
global global_info; % user data
global_info.STOP_AT = 200;
global_info.DELTA_TIME = 1;
global_info.cr = {'C_cr_3','C_cr_5','C_cr_2','C_cr_4','C_cr_6'};
%==================petri net structure=====================
pns = pnstruct('Example_pdf'); % create petri net structure
dyn.m0 = {'p1',5, 'p5',1, 'p6',1, 'vs1',1}; % Initial tokens of petri net
dyn.ft = {'allothers',0.000001,'tr_3',10,'tr_2',10,'tr_4',10,'tr_5',10,'tr
prnsys(pns, dyn);
pni = initialdynamics(pns, dyn);
%==================simulation results ======================
    sim = gpensim(pni); % perform simulation runs
    prnss(sim); % print the simulation results
%==================prncolormap results ======================
    prnfinalcolors(sim); % print only colors of the tokens that are presen
    prncolormap(sim, {'pcombined','p1', 'p2', 'p3', 'p4', 'p5', 'p6'});  %
%==================Transition utilization ======================
    duartion_martix = extractt(sim, {'t1','t2', 't3', 't4','tf_3','tf_2',
    disp('Duartion Martix:');
    disp(duartion_martix);
    occupancy_martix = occupancy(sim, {'t1','t2', 't3', 't4','tf_3','tf_2'
    disp('Occupancy Martix:');
    disp(occupancy_martix);
```

**FIGURE 8.** Part of COMMON_PRE file of colored controlled unreliable Petri net model in Figure 3.

$p_i \in P_{RH}$, and only one common recovery place for all places in $P_{RH}$. Thus, the computational complexity of the proposed Policy 2 is O(2$y$). For instance, consider an unreliable marked $S^3PR$ net shown in Figure 2, by Definition 11, we have $P_{Ru}$= $\{p_5, p_6\}$, $H(p_5) = \{p_3\}$, and $H(p_6) = \{p_2, p_4\}$, $P_{RH}$= $\{p_2, p_3, p_4, p_5, p_6\}$, $|P_{RH}| = 5$, the computational complexity is O(3*5). While by the proposed Policy 2, the computational complexity is O(2*5), which means that it has minimal computational complexity compared with Definition 11 [30]. In general, the computational complexity of the proposed method has polynomial time complexity.

## IV. GPenSIM CODE AND VALIDATION

Implementing a Petri net model as shown in Figure 4 by using GPenSIM code usually results in three files: PDF, COMMON_PRE, and MSF. Figure 6 illustrates the proposed PDF file and describes the structure of Petri net model by defining the sets of places, transitions, and arcs. Figure 7 illustrates the proposed COMMON_PRE file and describes the conditions for the enabled failure and recovery transitions to start firing based on mean time to failure and mean time to repair, respectively.

Figure 8 displays the MSF file used to compute the utilization of robot and machine, the liveness of Petri net model, and the throughput. To test and validate the proposed GPenSIM code, the code is validated and compared with Definition 11 [30]. The simulation is run for 480 min. After running and simulating the Petri net model, the results of the MATLAB simulation can be summarized as follows.

Table 1 lists the results in terms of the number of recovery places, failure transitions, recovery transitions, recovery arcs, liveness, utilization of robot and machine, and throughput. It is observed that the proposed approach provides a colored controlled unreliable marked Petri net with only a single recovery place compared with five recovery places in Definition 11 [30]. In terms of the utilization of robot and machine and the throughput, both methods obtain the same values. Therefore, the developed method is validated and can give sufficiently accurate results so that it can be used for other cases.

**TABLE 1.** Performance comparison of some deadlock control policies.

| Parameter | By [30] | Proposed Method |
|---|---|---|
| No. recovery places | 5 | 1 |
| No. failure transitions | 5 | 5 |
| No. recovery transitions | 5 | 5 |
| No. recovery arcs | 10 | 10 |
| Liveness | Live | Live |
| Robot utilization % | 48.75 | 48.75 |
| Machine utilization % | 49.1667 | 49.1667 |
| Throughput | 58 | 58 |

**FIGURE 9.** Automated manufacturing system at King Saud University.

## V. CASE STUDY

Figure 9 shows the AMS considered in this article. This system is available in the laboratory of Computer Integrated Manufacturing (CIM) at King Saud University [6].

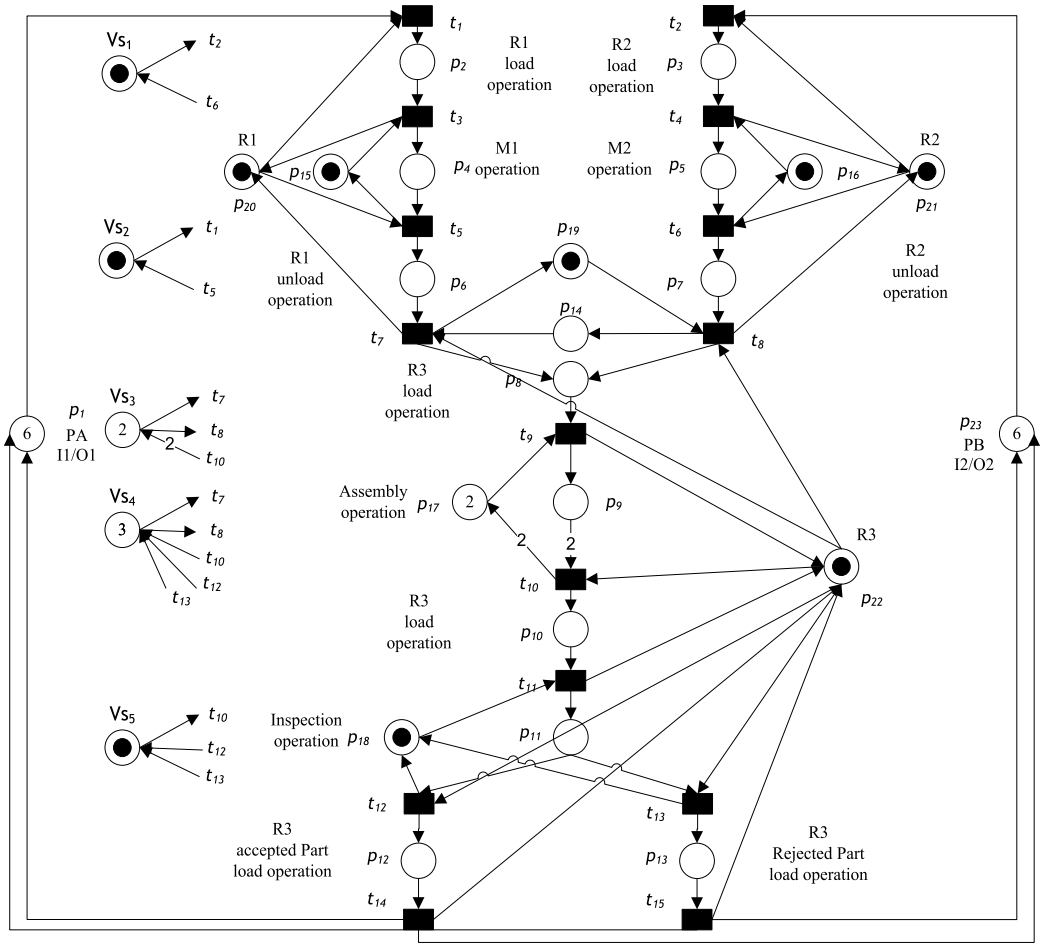


**FIGURE 10.** Petri net model of AMS.

It comprises input and output buffers, two M1-M2 machines, assembly and inspection stations, a conveyor to transfer the parts between machines and stations, and three robots R1–R3 for loading and unloading parts.

Each robot (machine) holds (processes) one part at the same time. Two part types A and B are considered to be processed in the system. The process route of part A and part B is constructed by M1 and M2, respectively. Then, two parts are assembled in an assembly station. The assembled part is inspected at an inspection station, and then the final product leaves the AMS.

Robots are working in a mutually exclusive manner. R1 loads/unloads part A on M1 from the conveyor, R2 loads/unloads part B into M2 from the conveyor, and R3 loads/unloads parts A and B onto the assembly and inspection stations from the conveyor. The Petri net model is illustrated in Figure 10. The Petri net model is composed of 15 transitions and 23 places. The places can be described as the following set partition: $P^0 = \{p_1, p_{23}\}$, $P_R = \{p_{15}, \ldots, p_{22}\}$, and $P_A = \{p_2, \ldots, p_{14}\}$. The properties of the developed Petri net models are obtained using the free GPenSIM tool [41]. It was found that the system is not live (deadlock).

The suggested deadlock prevention algorithm was applied to this case study. Without considering recovery subnets, the system model has five strict minimal siphons that can be empty: $S_1 = \{p_7, p_{16}, p_{21}\}$, $S_2 = \{p_6, p_{15}, p_{20}\}$, $S_3 = \{p_{10}, p_{12}, p_{13}, p_{17}, p_{22}\}$, $S_4 = \{p_{12}, p_{13}, p_{17}, p_{18}, p_{22}\}$, and $S_5 = \{p_8, p_{12}, p_{13}, p_{18}, p_{22}\}$. Based on the suggested deadlock prevention algorithm (Policy 1), five monitors are added to prevent the five strict minimal siphons from being emptied such that the original net without considering resource failures is controlled. The required control places using Policy 1 are designed as follows:

1. ${}^\bullet V_{S1} = \{t_6\}$, $V_{S1}^\bullet = \{t_2\}$, and $M_{Vo}(V_{S1}) = 1$
2. ${}^\bullet V_{S2} = \{t_5\}$, $V_{S2}^\bullet = \{t_1\}$, and $M_{Vo}(V_{S2}) = 1$
3. ${}^\bullet V_{S3} = \{2t_{10}\}$, $V_{S3}^\bullet = \{t_7, t_8\}$, and $M_{Vo}(V_{S3}) = 2$
4. ${}^\bullet V_{S4} = \{t_{10}, t_{12}, t_{13}\}$, $V_{S4}^\bullet = \{t_7, t_8\}$, and $M_{Vo}(V_{S4}) = 3$
5. ${}^\bullet V_{S5} = \{t_{12}, t_{13}\}$, $V_{S5}^\bullet = \{t_{10}\}$, and $M_{Vo}(V_{S5}) = 1$

Figure 10 shows the controlled Petri net model after the addition of the control places. In the net shown in Figure 9, we have $P_{Ru} = \{p_{15}, p_{16}, p_{17}, p_{18}, p_{20}, p_{21}, p_{22}\}$, $H(p_{15}) = \{p_4\}$, $H(p_{16}) = \{p_5\}$, $H(p_{17}) = \{p_9\}$, $H(p_{18}) = \{p_{11}\}$, $H(p_{20}) = \{p_2, p_6\}$, $H(p_{21}) = \{p_3, p_7\}$, and $H(p_{22}) = \{p_8, p_{10}, p_{12}, p_{13}\}$. Adding a common recovery subnet for $p_{15}$, $p_{16}$, $p_{17}$, $p_{18}$, $p_{20}$, $p_{21}$, and $p_{22}$ by Definition 12 results in the colored controlled unreliable Petri net model depicted in Figure 11, and Table 2 lists the NA, $T_F$, $T_R$, and $C_R$ of an unreliable Petri net.

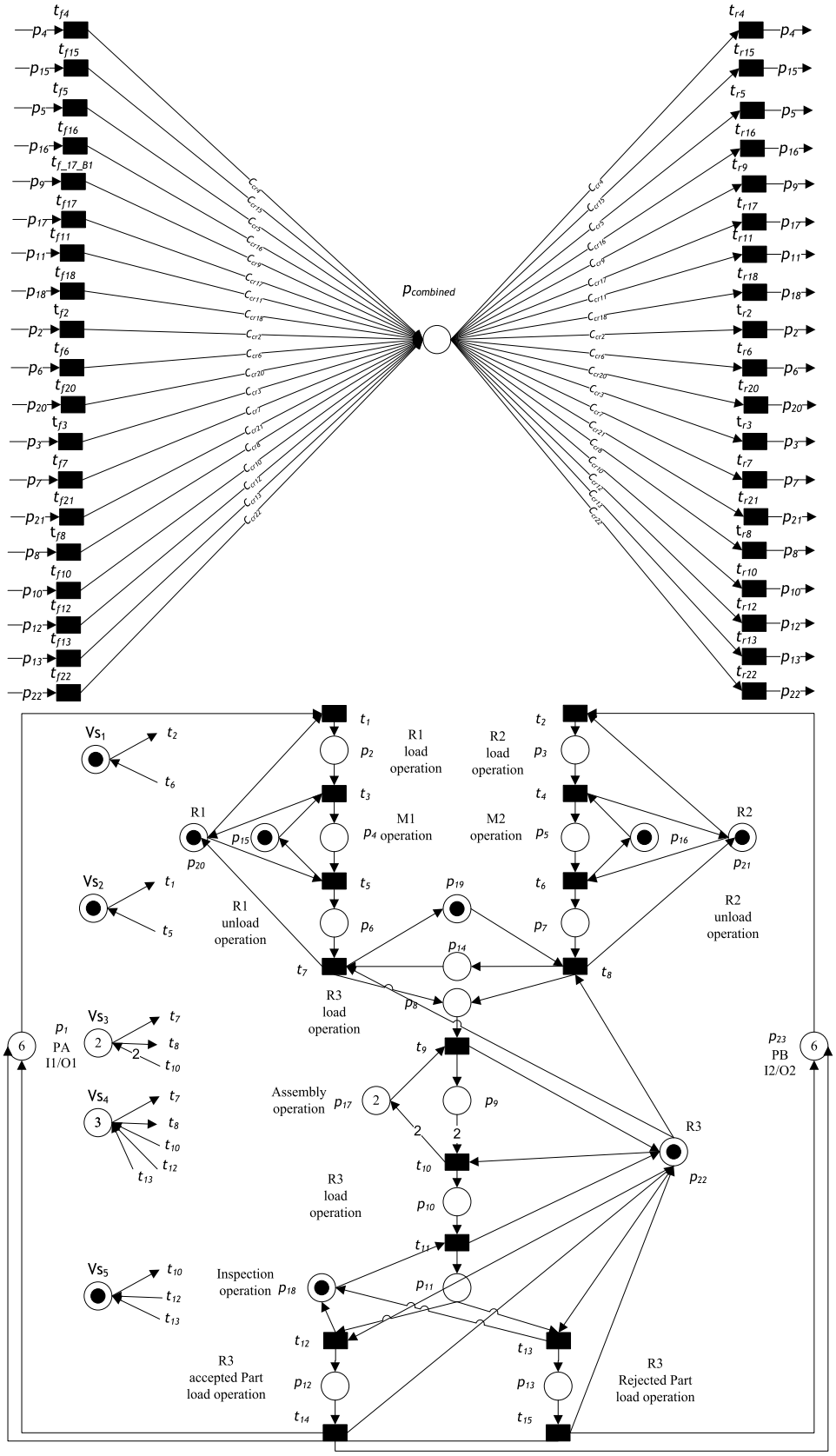


**FIGURE 11.** Colored controlled unreliable Petri net of AMS by Policy 2.

**TABLE 2.** PRu, NA, TF, TR, and CR of Unreliable Petri net.

| $P_{Ru}$ | NA | $T_F$ | $T_R$ | $C_R$ |
|---|---|---|---|---|
| $p_{15}$ | 4 | $t_{f4}$ | $t_{r4}$ | $C_{cr4}$ |
| | 15 | $t_{f15}$ | $t_{r15}$ | $C_{cr15}$ |
| $p_{16}$ | 5 | $t_{f5}$ | $t_{r5}$ | $C_{cr5}$ |
| | 16 | $t_{f16}$ | $t_{r16}$ | $C_{cr16}$ |
| $p_{17}$ | 9 | $t_{f9}$ | $t_{r9}$ | $C_{cr9}$ |
| | 17 | $t_{f17}$ | $t_{r17}$ | $C_{cr17}$ |
| $p_{18}$ | 11 | $t_{f11}$ | $t_{r11}$ | $C_{cr11}$ |
| | 18 | $t_{f18}$ | $t_{r18}$ | $C_{cr18}$ |
| $p_{20}$ | 2 | $t_{f2}$ | $t_{r2}$ | $C_{cr2}$ |
| | 6 | $t_{f6}$ | $t_{r6}$ | $C_{cr6}$ |
| | 20 | $t_{f20}$ | $t_{r20}$ | $C_{cr20}$ |
| $p_{21}$ | 3 | $t_{f3}$ | $t_{r3}$ | $C_{cr3}$ |
| | 7 | $t_{f7}$ | $t_{r7}$ | $C_{cr7}$ |
| | 21 | $t_{f21}$ | $t_{r21}$ | $C_{cr21}$ |
| $p_{22}$ | 8 | $t_{f8}$ | $t_{r8}$ | $C_{cr8}$ |
| | 10 | $t_{f10}$ | $t_{r10}$ | $C_{cr10}$ |
| | 12 | $t_{f212}$ | $t_{r12}$ | $C_{cr12}$ |
| | 13 | $t_{f13}$ | $t_{r13}$ | $C_{cr13}$ |
| | 22 | $t_{f22}$ | $t_{r22}$ | $C_{cr22}$ |

If resource $p_{15}$ fails in busy state $p_4$, i.e., $t_{f4}$ fires, it adds a color $C_{cr4}$ to the token from $p_4$ and deposits it into the common place $p_{combined}$. If resource $p_{15}$ fails in idle state $p_{15}$, i.e., $t_{f15}$ fires, then it adds a color $C_{cr15}$ to the token from $p_{15}$ and deposits it into the common place $p_{combined}$. After the failed resource is repaired $p_{15}$, the colored token in $p_{combined}$ flows into $p_4$ or $p_{15}$ by firing $t_{r4}$ or $t_{r15}$. When the transition $t_{r4}$ or $t_{r15}$ fires selects only the tokens with colors $C_{cr4}$ or $C_{cr15}$ from $p_{combined}$ and deposits it into $p_4$ or $p_{15}$, indicating that the recovery of resource $p_{15}$ is finished.

If resource $p_{16}$ fails in busy state $p_5$, i.e., $t_{f5}$ fires, it adds a color $C_{cr5}$ to the token from $p_5$ and deposits it into the common place $p_{combined}$. If resource $p_{16}$ fails in idle state $p_{16}$, i.e., $t_{f16}$ fires, it adds a color $C_{cr16}$ to the token from $p_{16}$ and deposits it into the common place $p_{combined}$. After the failed resource is repaired $p_{16}$, the colored token in $p_{combined}$ flows into $p_5$ or $p_{16}$ by firing $t_{r5}$ or $t_{r16}$. When transition $t_{r5}$ or $t_{r16}$ fires, selects only the tokens with colors $C_{cr5}$ or $C_{cr16}$ from $p_{combined}$ and deposits it into $p_5$ or $p_{16}$, indicating that the recovery of resource $p_{16}$ is finished.

If resource $p_{17}$ fails in busy state $p_9$, i.e., $t_{f9}$ fires, it adds a color $C_{cr9}$ to the token from $p_9$ and deposits it into the common place $p_{combined}$. If resource $p_{17}$ fails in idle state $p_{17}$, i.e., $t_{f17}$ fires, it adds a color $C_{cr17}$ to the token from $p_{17}$ and deposits it into the common place $p_{combined}$. After the failed resource is repaired $p_{17}$, the colored token in $p_{combined}$ flows into $p_9$ or $p_{17}$ by firing $t_{r9}$ or $t_{r17}$. When the transition $t_{r9}$ or $t_{r17}$ fires, selects only the tokens with colors $C_{cr9}$ or $C_{cr17}$ from $p_{combined}$ and deposits it into $p_9$ or $p_{17}$, indicating that the recovery of resource $p_{17}$ is finished.

If resource $p_{18}$ fails in busy state $p_{11}$, i.e., $t_{f11}$ fires, it adds a color $C_{cr11}$ to the token from $p_{11}$ and deposits it into the common place $p_{combined}$. If resource $p_{18}$ fails in idle state $p_{18}$, i.e., $t_{f18}$ fires, it adds a color $C_{cr18}$ to the token from $p_{18}$ and deposits it into the common place $p_{combined}$. After the failed resource is repaired $p_{18}$, the colored token in $p_{combined}$ flows into $p_{11}$ or $p_{18}$ by firing $t_{r11}$ or $t_{r18}$. When the transition $t_{r11}$ or $t_{r18}$ fires, selects only the tokens with colors $C_{cr11}$ or $C_{cr18}$ from $p_{combined}$ and deposits it into $p_{11}$ or $p_{18}$ indicating that the recovery of resource $p_{18}$ is finished.

If resource $p_{20}$ fails in busy state $p_2$ or $p_6$ i.e., $t_{f2\ or\ f6}$ fires, it adds a color $C_{cr2}$ or $C_{cr6}$ on the token from $p_2$ or $p_6$ and deposits it into the common place $p_{combined}$. If resource $p_{20}$ fails in idle state $p_{20}$ i.e., $t_{f20}$ fires, it adds a color $C_{cr20}$ on the token from $p_{20}$ and deposits it into the common place $p_{combined}$; after the failed resource is repaired $p_{20}$, the colored token in $p_{combined}$ flows into $p_2$, $p_6$ or $p_{20}$ by firing $t_{r2}$, $t_{r6}$ or $t_{r20}$, when the transition $t_{r2}$, $t_{r6}$ or $t_{r20}$ fires selects only the token with color $C_{cr2}$, $C_{cr6}$ or $C_{cr20}$ from $p_{combined}$ and deposits it into the $p_2$, $p_6$ or $p_{20}$ indicating that a resource $p_{20}$ recovery is finished.

If resource $p_{21}$ fails in busy state $p_3$ or $p_7$ i.e., $t_{f3\ or\ f7}$ fires, it adds a color $C_{cr3}$ or $C_{cr7}$ on the token from $p_3$ or $p_7$ and deposits it into the common place $p_{combined}$. If resource $p_{21}$ fails in idle state $p_{21}$ i.e., $t_{f21}$ fires, it adds a color $C_{cr21}$ on the token from $p_{21}$ and deposits it into the common place $p_{combined}$; after the failed resource is repaired $p_{21}$, the colored token in $p_{combined}$ flows into $p_3$, $p_7$ or $p_{21}$ by firing $t_{r3}$, $t_{r7}$ or $t_{r21}$, when the transition $t_{r3}$, $t_{r7}$ or $t_{r21}$ fires selects only the tokens with color $C_{cr3}$, $C_{cr7}$ or $C_{cr21}$ from $p_{combined}$ and deposits it into the $p_3$, $p_7$ or $p_{21}$ indicating that a resource $p_{21}$ recovery is finished.

Finally, if resource $p_{22}$ fails in busy state $p_8$, $p_{10}$, $p_{12}$ or $p_{13}$ i.e., $t_{f8}$, $t_{f10}$, $t_{f12}$, or $t_{f13}$ fires, it adds a color $C_{cr8}$, $C_{cr10}$, $C_{cr12}$, or $C_{cr13}$ on the token from $p_8$, $p_{10}$, $p_{12}$ or $p_{13}$ and deposits it into the common place $p_{combined}$. If resource $p_{22}$ fails in idle state $p_{22}$ i.e., $t_{f22}$ fires, it adds a color $C_{cr22}$ on the token from $p_{22}$ and deposits it into the common place $p_{combined}$; after the failed resource is repaired $p_{22}$, the colored token in $p_{combined}$ flows into $p_8$, $p_{10}$, $p_{12}$ or $p_{13}$ by firing $t_{r8}$, $t_{r10}$, $t_{r12}$, or $t_{r13}$, when the transition $t_{r8}$, $t_{r10}$, $t_{r12}$, or $t_{r13}$ fires selects only the tokens with color $C_{cr8}$, $C_{cr10}$, $C_{cr12}$, or $C_{cr13}$ from $p_{combined}$ and deposits it into the $p_8$, $p_{10}$, $p_{12}$, $p_{13}$, or $p_{22}$ indicating that a resource $p_{22}$ recovery is finished.

## VI. CONCLUSION

Various studies on the prevention of deadlocks have been devoted to enforcing liveness on the premise that all resources in a system work correctly. However, in practice, AMSs may face unexpected resource failures. This research suggested a robust deadlock-prevention controller for an AMS with all unreliable resources. In the absence or existence of resource failure, the proposed robust controller can ensure the required properties, i.e., the liveliness of the controlled system.

A two-step policy was developed in this paper to address the robust deadlock control of AMSs. The strict minimal siphon-based policy in [39] was utilized in the first step for a specific plant, resulting in a controlled net without

considering resource failures. Then, resource failures were considered by adding a common recovery subnet in a unified modeling method using a Petri net. The controlled net can work smoothly even if unreliable resources fail as long as there is no failure of one unit of unreliable resource.

The main advantages of the proposed method are: (1) It can be applied to an unreliable complex Petri net model for AMSs; (2) It has less computational complexity for the computation of the common recovery subnet; (3) It can obtain one common recovery subnet to model all resource failures; (4) It has a simpler structure compared to the technique used in [30]; (5) It does not need to compute reachability graphs compared to the policy developed in [35], which means that it has small computational overhead; (6) It can easily handle robust deadlock control problems with all unreliable resources without using inhibitor arcs; (7) Simulation, validation, and performance comparison are provided to compare the performance of the proposed method by using developed GPenSIM code.

The main limitation of the proposed method is that the proposed model may undergo changes of control specifications and requirements such as the processing routes of the system are changed, addition of new product, and addition of new machine. If a system has these issues, a system requires to be reconfigurable. Then the proposed model can have deadlocks. Therefore, our future research will investigate the proposed method to improve efficiency for valid and rapid reconfiguration of Petri net-based supervisory controllers for reconfigurable manufacturing systems.

## ACKNOWLEDGMENT

The authors would like to thank King Saud University for funding and supporting this research through Researchers Supporting Project Number (RSP-2019/62).

## REFERENCES

[1] Z. Wu Li, M. Chu Zhou, and N. Qi Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 173–188, Mar. 2008.

[2] Z. Li, N. Wu, and M. Zhou, "Deadlock control of automated manufacturing systems based on Petri nets—A literature review," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 437–462, Jul. 2012.

[3] A. M. El-Tamimi, E. A. Nasr, A. Al-Ahmari, H. Kaid, and Z. Li, "Evaluation of deadlock control designs in automated manufacturing systems," in *Proc. Int. Conf. Ind. Eng. Oper. Manage. (IEOM)*, Mar. 2015, pp. 1–10.

[4] Y. Chen, Z. Li, K. Barkaoui, and A. Giua, "On the enforcement of a class of nonlinear constraints on Petri nets," *Automatica*, vol. 55, pp. 116–124, May 2015.

[5] E. Abouel Nasr, A. M. El-Tamimi, A. Al-Ahmari, and H. Kaid, "Comparison and evaluation of deadlock prevention methods for different size automated manufacturing systems," *Math. Problems Eng.*, vol. 2015, pp. 1–19, 2015.

[6] H. Kaid, A. Al-Ahmari, A. M. El-Tamimi, E. A. Nasr, and Z. Li, "Design and implementation of deadlock control for automated manufacturing systems," *South Afr. J. Ind. Eng.*, vol. 30, no. 1, pp. 1–23, 2019.

[7] M. Bashir and L. Hong, "Global supervisory structure for decentralized systems of flexible manufacturing systems using Petri nets," *Processes*, vol. 7, no. 9, p. 595, Sep. 2019.

[8] Y. Chen, Z. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 2, pp. 374–393, Apr. 2011.

[9] A. Ghaffari, N. Rezg, and X. Xie, "Design of a live and maximally permissive Petri net controller using the theory of regions," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 137–142, Feb. 2003.

[10] M. Uzam and M. Zhou, "Iterative synthesis of Petri net based deadlock prevention policy for flexible manufacturing systems," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2004, pp. 4260–4265.

[11] M. Uzam, "The use of the Petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 23, nos. 3–4, pp. 204–219, Feb. 2004.

[12] D. Sun, Y. Chen, M. A. El-Meligy, M. A. F. Sharaf, N. Wu, and Z. Li, "On algebraic identification of critical states for deadlock control in automated manufacturing systems modeled with Petri nets," *IEEE Access*, vol. 7, pp. 121332–121349, 2019.

[13] D. Y. Chao, "Direct minimal empty siphon computation using MIP," *Int. J. Adv. Manuf. Technol.*, vol. 45, nos. 3–4, pp. 397–405, Nov. 2009.

[14] D. Y. Chao, "Improvement of suboptimal siphon-and FBM-based control model of a well-known $S^3PR$," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 2, pp. 404–411, Apr. 2011.

[15] W. Duan, C. Zhong, X. Wang, A. U. Rehman, U. Umer, and N. Wu, "A deadlock prevention policy for flexible manufacturing systems modeled with Petri nets using structural analysis," *IEEE Access*, vol. 7, pp. 49362–49376, 2019.

[16] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajuh, "Single controller-based colored Petri nets for deadlock control in automated manufacturing systems," *Processes*, vol. 8, no. 1, p. 21, Dec. 2019.

[17] 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. Technol.*, vol. 19, no. 3, pp. 192–208, Feb. 2002.

[18] D. Y. Chao, "Fewer monitors and more efficient controllability for deadlock control in S3 PGR 2 (systems of simple sequential processes with general resource requirements)," *Comput. J.*, vol. 53, no. 10, pp. 1783–1798, Dec. 2010.

[19] Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 34, no. 1, pp. 38–51, Jan. 2004.

[20] Y.-L. Pan, C.-Y. Tseng, and T.-C. Row, "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.*, vol. 11, no. 3, pp. 261–270, Sep. 2017.

[21] M. Zhao and M. Uzam, "A suboptimal deadlock control policy for designing non-blocking supervisors in flexible manufacturing systems," *Inf. Sci.*, vols. 388–389, pp. 135–153, May 2017.

[22] X. Cong, C. Gu, M. Uzam, Y. Chen, A. M. Al-Ahmari, N. Wu, M. Zhou, and Z. Li, "Design of optimal Petri net supervisors for flexible manufacturing systems via weighted inhibitor arcs," *Asian J. Control*, vol. 20, no. 1, pp. 511–530, Jan. 2018.

[23] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S 4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.

[24] Q. Zhuang, W. Dai, S. Wang, J. Du, and Q. Tian, "An MIP-based deadlock prevention policy for siphon control," *IEEE Access*, vol. 7, pp. 153782–153790, 2019.

[25] X. Guo, S. Wang, D. You, Z. Li, and X. Jiang, "A siphon-based deadlock prevention strategy for $S^3PR$," *IEEE Access*, vol. 7, pp. 86863–86873, 2019.

[26] M. Lawley and W. Sulistyono, "Robust supervisory control policies for manufacturing systems with unreliable resources," *IEEE Trans. Robot. Autom.*, vol. 18, no. 3, pp. 346–359, Jun. 2002.

[27] F.-S. Hsieh, "Robustness analysis of Petri nets for assembly/disassembly processes with unreliable resources," *Automatica*, vol. 42, no. 7, pp. 1159–1166, Jul. 2006.

[28] S. Wang, S. Foh Chew, and M. Lawley, "Using shared-resource capacity for robust control of failure-prone manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 38, no. 3, pp. 605–627, May 2008.

[29] S. Foh Chew, S. Wang, and M. Lawley, "Robust supervisory control for product routings with multiple unreliable resources," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 195–200, Jan. 2009.

[30] G. Liu, Z. Li, K. Barkaoui, and A. M. Al-Ahmari, "Robustness of deadlock control for a class of Petri nets with unreliable resources," *Inf. Sci.*, vol. 235, pp. 259–279, Jun. 2013.

[31] H. Yue, K. Xing, and Z. Hu, "Robust supervisory control policy for avoiding deadlock in automated manufacturing systems with unreliable resources," *Int. J. Prod. Res.*, vol. 52, no. 6, pp. 1573–1591, Mar. 2014.
[32] H. Yue, K. Xing, H. Hu, W. Wu, and H. Su, "Robust supervision using shared-buffers in automated manufacturing systems with unreliable resources," *Comput. Ind. Eng.*, vol. 83, pp. 139–150, May 2015.
[33] F. Wang, K.-Y. Xing, M.-C. Zhou, X.-P. Xu, and L.-B. Han, "A robust deadlock prevention control for automated manufacturing systems with unreliable resources," *Inf. Sci.*, vol. 345, pp. 243–256, Jun. 2016.
[34] Y. Feng, K. Xing, Z. Gao, and Y. Wu, "Transition cover-based robust Petri net controllers for automated manufacturing systems with a type of unreliable resources," *IEEE Trans. Syst., Man, Cybern., Syst*, vol. 47, no. 11, pp. 3019–3029, Nov. 2017.
[35] G. Liu, P. Li, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs," *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 49, no. 7, pp. 1371–1385, Jul. 2019.
[36] N. Ran, J. Hao, Z. Dong, Z. He, Z. Liu, Y. Ruan, and S. Wang, "K-codiagnosability verification of labeled Petri nets," *IEEE Access*, vol. 7, pp. 185055–185062, 2019.
[37] X. Li, G. Liu, Z. Li, N. Wu, and K. Barkaoui, "Elementary siphon-based robust control for automated manufacturing systems with multiple unreliable resources," *IEEE Access*, vol. 7, pp. 21006–21019, 2019.
[38] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer-Verlag, 2009.
[39] J. Ezpeleta, J. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.
[40] Y. Chen, Z. Li, and M. Zhou, "Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 3, pp. 615–629, May 2012.
[41] R. Davidrajuh, *Modeling Discrete-Event Systems with GPenSIM: An Introduction*. Cham, Switzerland: Springer, 2018.

**ABDULRAHMAN AL-AHMARI** received the Ph.D. degree in manufacturing systems engineering from the University of Sheffield, Sheffield, U.K., in 1998. He worked as the Dean of the Advanced Manufacturing Institute, Chairman of Industrial Engineering Department. He is currently a Professor of industrial engineering with King Saud University, Riyadh, Saudi Arabia. He led a number of funded projects from different organizations in Saudi Arabia. He has published papers in leading *Journal of Industrial and Manufacturing Engineering*. His current research interests include advanced manufacturing technologies, Petri nets, analysis and design of manufacturing systems, computer integrated manufacturing, optimization of manufacturing operations, flexible manufacturing systems and cellular manufacturing systems, and applications of decision support systems in manufacturing.

**HUSAM KAID** received the B.S. degree in industrial engineering from the University of Taiz, Taiz, Yemen, in 2010, and the M.S. degree in industrial engineering from King Saud University, Saudi Arabia, in 2015. He is currently a Researcher and a Ph.D. Student with the Industrial Engineering Department, College of Engineering, King Saud University, Saudi Arabia. His research areas and specialties are design and analysis of manufacturing systems, deadlock control in manufacturing systems, supply chain, simulation, operations research, optimization techniques, and bibliometric network analysis.

**ZHIWU LI** (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in mechanical engineering, automatic control, and manufacturing engineering from Xidian University, Xi'an, China, in 1989, 1992, and 1995, respectively. He joined Xidian University, in 1992. Over the past decade, he was a Visiting Professor with the University of Toronto, Technion (Israel Institute of Technology), Martin-Luther University, Conservatoire National des Arts et Métiers (Cnam), Meliksah Universitesi, University of Cagliari, University of Alberta, and King Saud University. He is currently with the Macau University of Science and Technology. His current research interests include Petri net theory and application, supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining. He is listed in Marquis Who's Who in the world, 27th Edition, 2010. He was a recipient of an Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation, Germany, and Research in Paris, France. He is the Founding Chair of the Xi'an Chapter of IEEE Systems, Man, and Cybernetics Society. He serves as a reviewer for more than 90 international journals.

**REGGIE DAVIDRAJUH** (Senior Member, IEEE) received the master's degree in control systems engineering and the Ph.D. degree in industrial engineering from the Norwegian University of Science and Technology (NTNU), in 1993 and 2001, respectively, and the D.Sc. (Dr hab) degree in informatics from the AGH University of Science and Technology, Poland, in 2016. He is currently a Professor of informatics with the Department of Electrical and Computer Engineering, University of Stavanger, Norway. He is also a Visiting Professor with the Silesian University of Technology, Poland. His current research interests include discrete-event systems, Petri nets, and graph algorithms.