IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 12, 2021, accepted July 3, 2021, date of publication July 12, 2021, date of current version July 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096595

Petri Net-Based Robust Supervisory Control of Automated Manufacturing Systems With Multiple Unreliable Resources

UMAR SULEIMAN ABUBAKAR[1], GAIYUN LIU[1], (Senior Member, IEEE), AND MURAT UZAM[2]

[1]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[2]Elektrik-Elektronik Mühendisliği Bölümü, Yozgat Bozok Üniversitesi, 66900 Yozgat, Turkey

Corresponding author: Gaiyun Liu (gaiyunliu@gmail.com)

This work was supported in part by the Natural Science Foundation of Shaanxi Province under Grant 2019JM-049, and in part by the National Natural Science Foundation of China under Grant 61873342 and Grant 61304051.

ABSTRACT In recent years, there has been a growing concern for robust supervisory control policies that can handle both deadlock and blockage propagation in automated manufacturing systems in the event of resource failures. This work proposes a novel robust supervisory control policy for automated manufacturing systems with multiple unreliable resources without using central buffers. The policy permits legal states as many as possible and ensures that parts not requiring unreliable resources can be automatically processed without human intervention if one or multiple unreliable resources fail. It is based on the modified neighborhood policy, namely single route neighborhood, which handles the allocation of failure-prone resources in a system. To guarantee deadlock-free operations in the remaining parts of the system, monitors are designed for emptiable strictly minimal siphons. Through examples, the applicability of the proposed policy is demonstrated.

INDEX TERMS Petri net, deadlock, automated manufacturing system, robust supervisory control.

I. INTRODUCTION

Automated manufacturing systems (AMSs) are resource allocation systems whose components or parts interact with one another while competing for limited reusable resources. A resource allocation problem deals with allocating limited resources available among a number of processes within a system. Optimal distribution and sharing of these limited resources are desired in an AMS. This optimality of distributing resources could be stymied if there is no effective and robust supervisory control in place to deal with situations that may arise in the system such as deadlocks and blockage [1], [2].

A plethora of studies in the literature deal with deadlock problems, and substantial achievements have been made in this regard. Thus far, a vast majority of them are based on the premise that the resources are all reliable [3]–[25], [52], [53], [57]–[59]. Contrary to this notion of absolute resource reliability, in reality, resources do fail or break down due to aging or wear and tear, faults in some resource's components or due to failure of sensors [26]. Furthermore, despite putting in place an effective policy to resolve deadlock in a system, resource failures could still bring about deadlock and blockage issues [27]. A circular wait can be created due to the failure of a resource, which could stall the production of other part types that do not require the services of the failed resources.

Awakened by this reality, researchers have proposed a number of ideas of robust supervisory control for AMSs. One of the early and significant contribution is the work [2]. In [2], a system has only one unreliable resource. By combining a neighborhood policy and resource order policy, a robust supervisory control is developed. The neighborhood policy requires the use of inequality constraints, known as neighborhood constraints (NHC). This idea is later extended in [26] to handle failures in a system with multiple unreliable resources. Two supervisory control policies are formulated: one using NHC and the banker's algorithms and another using the banker's algorithm and single-step look ahead policy.

The study [28] proposes a policy that comprises the concept of remaining resource capacity constraints and a modified banker's algorithm to deal with AMS type

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

in [26]. The policy is more permissive than that in [26]. Wang *et al.* [29] use shared resource capacity and classify resources into three regions: region of failure dependence, that of continuous operation and that of distribution. In each region, a set of inequality constraints is computed. Two policies for robust supervisory control are developed. Chew *et al.* [30] present two robust supervisory control policies for AMSs with multiple unreliable resources and parts having more than one unreliable resources in their processing routes. The first policy is composed of NHC in conjunction with the banker's algorithm, while the second one comprises NHC and single-step look ahead. Either of them partitions parts processing routes into subroutes, and utilizes central buffers to store parts temporarily if an unreliable resource fails. Despite being more permissive than the policies in [26], the requirement for cental buffers, which means additional hardware, has made the policies expensive to implement.

An idea of employing the banker's algorithm together with available resource constraints is presented in [31] to design a robust AMS controller. The available resource capacity constraints require that at least one buffer space between a pair of failure-dependent resources should remain for the execution of the banker's algorithm. Informally, a resource is said to be failure-dependent on an unreliable resource if every part that enters its buffer space requires future processing on that unreliable resource. An unreliable resource is failure-dependent on itself.

The work [32] incorporates NHC with a resource order to build a supervisory control policy in order to tackle simultaneous failures of multiple unreliable resources in AMSs. Yue *et al.* [33] present a new policy of deadlock avoidance by using shared buffers and first and second order banker's algorithms. Unlike the absorbing type policies in [2], [26], [28], the policy in [29], [32], [33] is distributed. The classifications of robust supervisory control policies into absorbing and distributive types are proposed in [34]. If an unreliable resource fails, with the first type policies, all parts that require the failed resource in their remaining routes are absorbed into the buffer spaces of failure-dependent resources. With the second type policies, all parts that require the failed resource in their remaining routes are distributed among the buffer spaces of shared resources.

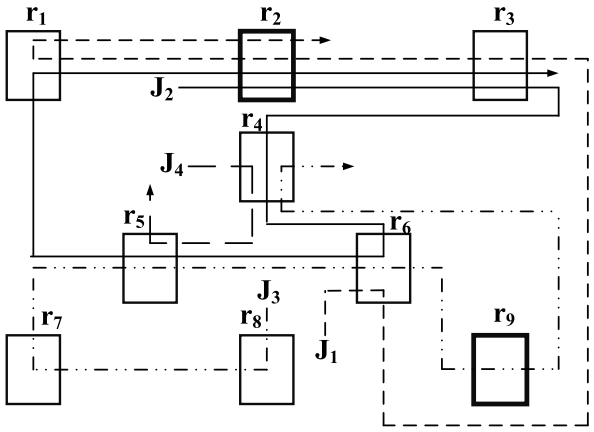
In [35]–[37], Hsieh proposes a number of methods to check the possibility of continuing production with a set of resource failures. The resource failures are modelled by using token extractions from a Petri net. His studies lay down fault tolerant conditions and propose a structural decomposition technique to check the feasibility of a production line. However, they are not intuitive to Petri net models [27]. Liu *et al.* [27] come up with the idea of adding recovery subnets and control places to a Petri net model of a system with unreliable resources. It is a robust control strategy that makes the controlled net live and prevents deadlock when some of the resources of the system fail. Other robust deadlock control policies proposed by Liu *et al.* can be found in [38]–[41].

In [42], Petri nets are used to model and develop a robust controller for deadlock prevention in AMSs. A Petri net model is divided into two subnets according to whether parts require unreliable resource to be processed or not. The subnet of parts that require unreliable resources is called a blocked net, while the other in which parts do not require unreliable resource is called a free subnet. A robust controller in [42] is composed of three layers. The first layer ensures the liveness of the system in the presence or absence of failures. The second one guarantees the processing of parts not requiring failed resources if there is a failure of an unreliable resource. The so-called second-level deadlocks caused by the application of the controllers are prevented by the third layer controller. Yue *et al.* [43] utilize both automata and Petri nets to model, analyze and design a supervisory control policy for deadlock/blockage avoidance. However, Petri nets are only used in modeling buffer net constraints.

Another robust supervisory control for AMSs that utilizes both Petri net and automaton to develop an AMS model is [44]. They propose a modified banker's algorithm which can be integrated with a deadlock avoidance policy for a system of simple sequential process with resources (S^3PR). For more details about the studies found in this area of robust supervisory control of AMSs, the reader can refer to [45]–[50] and a recent survey [51].

Owing to their superior structural characteristics, more compacted structure and more powerful capability to model and analyze complex systems, Petri nets are more effective in modeling, detecting and solving deadlocks. For instance, Petri nets can model systems with more sophisticated behaviors such as flexible routines, assembly and deassembly operations and multi-type as well as multi-quantity of resources and their combinations. Automata are more suitable for modeling simple systems such as single-unit resource allocation systems. Thus, automata are limited in their capability for modeling, simulation and supervisory control of AMSs [49].

NHC proposed in [2] has been used in a number of studies in conjunction with other policies to design robust supervisory control policies using automata. The idea has come across as quite intuitive to Petri net models. It can thus be implemented in Petri net models. However, it is too restrictive, and may prevent markings that cause neither deadlock nor blockage propagation in Petri net models from being reached if an unreliable resource fails. By taking advantage of Petri nets' superior structural characteristics, this work attempts not only to use NHC in a Petri net model, but also to propose new concepts that can help improve it in terms of behavioral permissiveness and structural complexity while achieving robust control. The study involves distributions of part type stages in order to avoid deadlocks and blockage when unreliable resources fail. As mentioned in [51], almost all the robust supervisory prevention strategies synthesized by adding control places



Set of all resources: $P_R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$
Set of reliable resources: $P^r_R = \{r_1, r_3, r_4, r_5, r_6, r_7, r_8\}$
Set of unreliable resources $P^u_R = \{r_2, r_9\}$
Set of processes or jobs $J = \{J_1, J_2, J_3, J_4\}$
Buffer spaces: $B = \langle B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9\rangle = \langle 1, 1, 1, 2, 1, 1, 1, 1, 1\rangle$
Set of part type stages of $J_1 = \{p_{11}, p_{12}, p_{13}, p_{14}, p_{15}\}$
Route of J_1: $\Gamma_1 = \langle r_6, r_3, r_2, r_1, r_2\rangle$
Set of part type stages of $J_2 = \{p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}, p_{27}, p_{28}\}$
Route of J_2: $\Gamma_2 = \langle r_2, r_3, r_4, r_6, r_5, r_1, r_2, r_3\rangle$
Set of part type stages of $J_3 = \{p_{31}, p_{32}, p_{33}, p_{34}, p_{35}, p_{36}\}$
Route of J_3: $\Gamma_3 = \langle r_8, r_7, r_5, r_6, r_9, r_4\rangle$
Set of part type stages of $J_4 = \{p_{41}, p_{42}\}$
Route of J_4: $\Gamma_4 = \langle r_4, r_5\rangle$

FIGURE 1. AMS with unreliable resources.

are based on the system in which each resource is modelled as several machines and robots, such as AMSs in [27], not as a workstation with some buffer spaces. Therefore, it is necessary to do more research to develop robust supervisory controllers for the type of AMS modelled as workstations with buffer spaces, which are implemented by adding control places to the Petri net model of the AMS. This work aims to make the following contributions:

1) Proposing a robust supervisory control policy based on NHC to prevent both deadlocks and blockages in AMSs using Petri nets, which is more permissive than NHC, and can permit as many makings as possible and prevents those that could cause deadlocks or blockages when unreliable resources fail;
2) Developing a policy that can handle systems with multiple unreliable resources in which part type stages require multiple unreliable resources in their processing routes without using a central buffer, which NHC alone cannot handle; and
3) Introducing inseparable constraints and neighborhood cluster concepts in order to improve the permissiveness of NHC and to reduce the structural complexity of the controlled Petri net model. The proposed policy consists of two strategies: single route neighborhood constraints (SNCs) and siphon-based deadlock control policy.

The rest of this paper is arranged as follows. Section II delineates AMSs with unreliable resources and their characteristics. Section III gives the motivations of this study. The new design of robust supervisory controllers is presented in Section IV. Section V gives the discussions and comparison between this work and previous work and Section VI concludes this study.

II. AMS WITH UNRELIABLE RESOURCES

We use the same AMS as that in [26] and maintain the same properties of the AMS as an example to present our ideas. Fig. 1 shows an AMS as a single unit resource allocation system with nine resources, where r_2 and r_9 are unreliable resources. The capacity of the buffer of each resource, the stages and routes of every part-type are indicated. The Petri net model of the AMS is an S^3PR subclass of Petri nets. The basics of Petri nets and some definitions, properties of S^3PRs, and basics of elementary siphons can be found in Section I of a supplementary file of this paper.

Petri nets are used to model AMS in such a way that only part movements among buffer spaces are controllable. As a result, only buffer spaces and the movements among them are modelled in the Petri net model. Therefore, all resources or workstations are buffer spaces and the Petri net is called a buffer net [48]. In this case, resource allocation and deallocation mean allocation and deallocation of buffer spaces. Fig. 2 depicts the S^3PR of the AMS. The system is a marked buffer net. The initial markings of the idle places are also indicated. The idle places are implicit places [19] such that the initial number of tokens in each idle place is equal to the total capacity of the resources used by the process. The initial marking of a resource represents its buffer capacity.

Let $(N_u, M_{u0}) = (P_A \cup P^0 \cup P_R, T, F, M_{u0})$ be the buffer net of the unreliable AMS with a set of resources $P_R = P^r_R \cup P^u_R$ with $P^r_R \cap P^u_R = \emptyset$, where P^r_R is the set of reliable resources, P^u_R is the set of unreliable resources, and P_A is the set of activity or operation places. Let $B = \{B_i : i = 1, 2, \ldots, |P_R|\}$ be the set of buffers associated with every resource type of the system $r_i \in P_R$. We assume that each resource of the system has buffer capacity B_i to accommodate any part type stage, both its finished units and unfinished units that need to be processed in the workstation. Let P^0 be the set of part types processed by the system, also known as the set of idle places in the Petri net model. Each part type $p_j \in P^0$ has an n-ordered set of processing stages $P_j = \{p_{j1}, p_{j2}, \ldots, p_{jn}\}$, where $n = |P_j|$. Each part type stage, also called part instance, is represented in the Petri net model as an operation place or activity place, p_{jk}, representing the k-th processing stage of part type instance of p_j and its set is P_A. Let J_j represent the whole process of processing a part type p_j in the production route Γ_j and let the loading/unloading of buffers be I_j/O_j. Therefore, the route of p_j is $\Gamma_j = \langle R(p_{j1}), R(p_{j2}), \ldots, R(p_{j|P_j|})\rangle$. Let $R : p_{jk} \rightarrow r_i$ such that $R(p_{jk}) = r_i$ is the case where p_{jk} requires an instance of resource r_i and

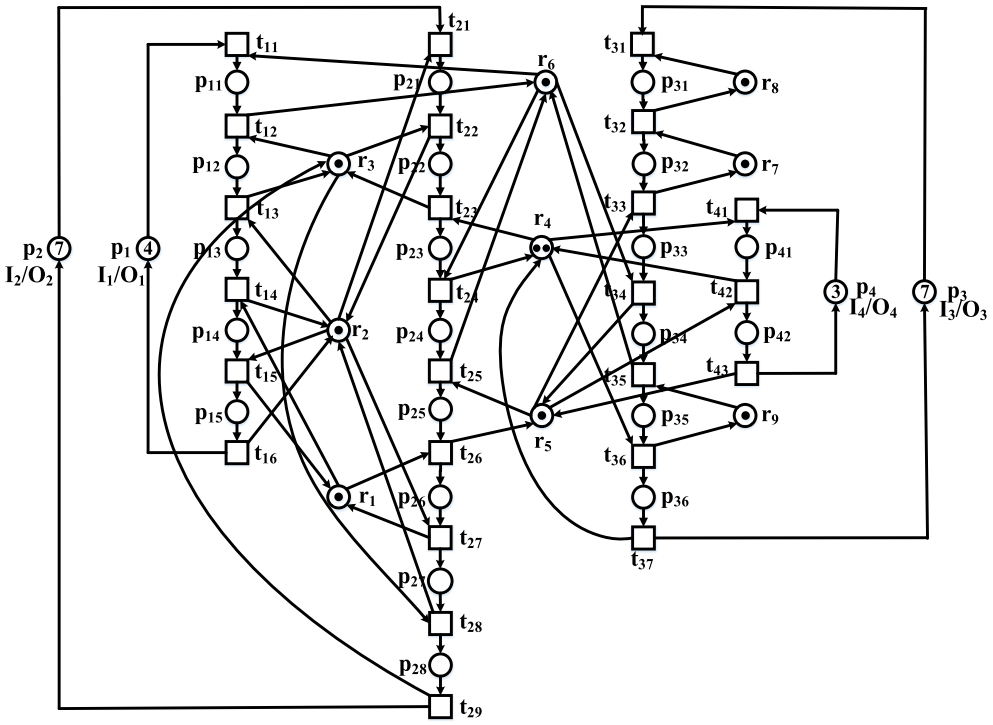


FIGURE 2. Petri net model of the AMS.

$R(p_{jk})$ returns resource required by p_{jk}. Let $H(r_i) = \{p_{jk} \in P_A | R(p_{jk}) = r_i\}$ be the set of operation places supported by r_i and each $p_{jk} \in H(r_i)$ is supported by an instance of r_i to execute its operation.

We assume that each resource type is a workstation composed of buffer spaces that accommodate and hold parts to be processed, and a server or processor that processes parts occupying the buffer spaces. A resource failure means the failure of a processor or any piece of equipment in the workstation that makes processing of parts impossible. We assume that the failure of a workstation does not affect its buffer space. As a result, we can continue to allot its buffer space provided that it is not full. However, the waiting parts in the buffer space cannot be processed until the fault in the workstation or server is fixed. Furthermore, the finished parts in the workstation can move away and continue along their respective routes to their next workstation or move out of the system if they are in their last processing stage. A workstation failure does not damage the part type stage that it is currently processing when a failure occurs, and a failure occurs only when its server is operating on parts [26].

III. MOTIVATION

The system in Fig. 1, without a robust controller, is in deadlock. Moreover, if we design a deadlock control policy without taking into account the unreliable nature of some resources in the system, in the event that one of the unreliable resources fails, the blocking effect as a result of the failure could propagate throughout the system and could result in a complete system failure. For instance, to observe such effect, consider the Petri net model of the AMS in Fig. 2. Firing the sequence of transitions $\sigma = t_{21}t_{22}t_{23}t_{24}t_{25}t_{31}t_{26}t_{11}t_{32}t_{27}$ from the initial marking M_0 leads to the marking $M = 3p_1 + 6p_2 + 6p_3 + 3p_4 + p_{11} + p_{27} + p_{32} + r_1 + r_3 + 2r_4 + r_5 + r_8 + r_9$. Apparently this is a safe state. However, if r_2 fails at M, the jobs J_1 and J_2 that require r_2, cannot be completed, and J_3 could be blocked from accessing r_6 by J_1 since J_1 is holding r_6 at p_{11}. If t_{12} fires at M after the failure of r_2 and J_1 moves from p_{11} to p_{12} after being processed by r_6, a new marking $M' = 3p_1 + 6p_2 + 6p_3 + 3p_4 + p_{12} + p_{27} + p_{32} + r_1 + 2r_4 + r_5 + r_6 + r_8 + r_9$ is generated. At M', J_3 can be completed since r_6 is marked. However, the system enters a deadlock state when r_2 is repaired since J_2 is currently holding r_2 at p_{27} and J_1 is holding r_3 at p_{12}. When r_2 is recovered and finishes processing J_2 at p_{27}, a situation is created in which J_2 is requesting r_3, $R(p_{28}) = r_3$ which is now being held by J_1 and J_1 is requesting r_2, $R(p_{13}) = r_2$ which is now occupied by J_2. This creates a deadlock state.

As another example, consider a reachable marking $M'' = 4p_1 + 3p_2 + 4p_3 + 2p_4 + p_{22} + p_{25} + p_{26} + p_{27} + p_{31} + p_{34} + p_{35} + p_{41} + r_3 + r_4 + r_7$. The job J_3 halts completely, and both J_3 and J_4 are completely blocked if r_2 fails at M''. The failure of r_9 brings the job J_3 to a standstill, and J_1 and J_2 will be blocked. We therefore need a robust supervisor to deal with such undesired situations. The following are the properties that such a supervisor should satisfy [26]:

1) A supervisor guarantees continued production of part types that do not require failed resources, given the absence of additional resource failures/repairs in the system.

2) A supervisor ensures continued production of all part types that do not require failed resource if an additional resource failure occurs.
3) A supervisor ensures continued production of all part types that do not require failed resource if a failed resource is repaired and put back to operation.

IV. ROBUST LIVENESS-ENFORCING SUPERVISOR

In this section, we present a supervisory control policy that satisfies the properties stated in Section III. The control policy is a combination of part types flow constraints within failure dependent resources (SNC) and siphon-based deadlock control method. SNC imposes restrictions on the distribution of part types that require unreliable resources within their respective failure-dependent resources, while the siphon-based method is achieved by designing monitors for unmarked minimal siphons found using a mixed integer programming (MIP) method to prevent deadlock in the remaining parts of the system that do not require unreliable resources. A set of inequality constraints are derived based on SNC and added to the unreliable Petri net model in the form of control places. The Petri net model obtained as a result of adding the constraints to the original Petri net model of the AMS is called a constrained or reduced Petri net model of the original one. Note that in this paper we use the terms constrained Petri net model and reduced Petri net model interchangeably. Though the policies of this paper do not require the use of a reachability graph, however, some concepts related to it are helpful to develop the techniques to be used in the control policy. These concepts can be found in Section II of the supplementary file of this paper.

Let the set of legal markings in a Petri net model (N_u, M_{u0}) of a system with a set of unreliable resources P_R^u be $\mathcal{M}_L$ and the set of deadlock markings be $\mathcal{M}_B$. Designing monitors as control places to prevent deadlock markings in $\mathcal{M}_B$ from being reached and allow all markings in $\mathcal{M}_L$ of (N_u, M_{u0}) to be reached cannot guarantee a robust supervisory control. This is because there may exist markings in $\mathcal{M}_L$ that can lead to deadlock or blockage, if an unreliable resource fails. Let the set of the markings that can lead to deadlock or blockage, if an unreliable resource fails, be denoted by $\mathcal{M}_F$. For robust control, since $\mathcal{M}_F \subseteq \mathcal{M}_L$ and they can lead to deadlock or blockage if unreliable resources in the system fail, the set of legal markings $\mathcal{M}_L$ should not contain such markings. Let $\mathcal{M}_{Lu} = \mathcal{M}_L \setminus \mathcal{M}_F$ be the set of robust legal markings of the Petri net model of a system with unreliable resources (N_u, M_{u0}), and $\mathcal{M}_{Du} = \mathcal{M}_B \cup \mathcal{M}_F$ be the set of illegal markings of its Petri net model.

A. SINGLE-ROUTE NEIGHBORHOOD OF FAILURE-DEPENDENT RESOURCES

For single-route neighborhood, we consider one route at a time and construct the neighborhood sets associated with every unreliable resource along that route. We repeat the same procedure for all the routes that contain unreliable resources. After generating single-route neighborhood for every route with unreliable resources, we then construct SNC in the form of inequalities to impose restrictions on the parts flow along these routes. We first compute SNC for every single route. After that we consider routes that overlap, i.e., routes that have shared failure-dependent resources (the definition of a failure-dependent resource is given later in this section) and construct constraints to resolve conflicts between the routes. Since resource failure is a stochastic event, our objective is to derive constraints to control the flows of parts that require unreliable resources without suppressing too many legal markings that does not lead to deadlock or blockage in a system if a random failure of an unreliable resource occurs.

Definition 1: Let $(N_u, M_{u0}) = (P_A \cup P^0 \cup P_R, T, F)$ be a marked buffer net model of an AMS with a set of unreliable resources P_R^u. Let $r_i \in P_R^u$. $r_d \in P_R$ is said to be failure-dependent on r_i if every part that enters the buffer space of r_d requires future processing on r_i. For $r_i \in P_R^u$, let $R_f(r_i)$ be the set of resource failure-dependent on r_i. Then $R_f(r_i) = \{r_d | r_d \in P_R, \forall p_{jk} \in H(r_d), \exists p_{j(k+c)} \in H(r_i), c \geq 0\}$ is the set of failure-dependent resources on r_i. $r_i \in P_R^u$ is apparently failure-dependent on itself, $r_i \in R_f(r_i)$. For $r_v, r_w \in P_R^u$ and $r_v \neq r_w$, $R_f(r_v) \cap R_f(r_w) = \emptyset$. We denote by $R_f = \cup_{r_i \in P_R^u} R_f(r_i)$ the set of failure-dependent resources and by $\tilde{\mathrm{R}}_f = P_R \setminus R_f$ the set of non-failure-dependent resources.

The AMS in Fig. 1 has two unreliable resources, $P_R^u = \{r_2, r_9\}$. Whenever r_1 appears in a route, r_2 appears later. This means that any part type processed by r_1 will require future processing on r_2. This makes r_1 a failure-dependent resource on r_2. Therefore, $R_f(r_2) = \{r_1, r_2\}$ since r_2 is failure-dependent on itself. Likewise any part type processed by r_7 and r_8 will require future service of r_9. Accordingly, $R_f(r_9) = \{r_7, r_8, r_9\}$. Thus, we have $R_f = \{r_1, r_2, r_7, r_8, r_9\}$. For the system of Fig. 1, we have $\tilde{\mathrm{R}}_f = \{r_3, r_4, r_5, r_6\}$ with $R_f \cap \tilde{\mathrm{R}}_f = \emptyset$.

Definition 2: The single-route neighborhood of a failure-dependent resource $r_d \in R_f(r_i)$ in a given route Γ_j is the set of part type stages denoted by $\mathcal{N}_j^d$ that require the service of r_d at some point in their processing stages without any intermediate failure-dependent resource of $R_f(r_i)$

Recall that in the Petri net model of Fig. 2, the part type stages are represented by the operation places. In other words, for a given route Γ_j, $\mathcal{N}_j^d$ is the set of operation places currently holding r_d, $H_j(r_d)$ and those holding non-failure dependent resources that precede r_d in the route without any intervening failure-dependent resource. Therefore, $\mathcal{N}_j^d = H_j(r_d) \cup \{p_{jk} | \exists c > 0$ with $p_{j(k+c)} \in H_j(r_d) \wedge \forall b \in [0, c), R(p_{j(k+b)}) \notin R_f(r_d)\}$. For $r_v, r_w \in R_f$ and $r_v \neq r_w$, $\mathcal{N}_j^v \cap \mathcal{N}_j^w = \emptyset$. Let $\mathcal{N}_j = \{\mathcal{N}_j^d | r_d \in R_f\}$ be the set of single-route neighborhoods of failure-dependent resources in the route Γ_j and $\mathcal{N} = \{\mathcal{N}_j | r_d \in R_f\}$ be the set of all single-route neighborhoods of failure-dependent resources of the Petri net model of an unreliable AMS.

In the case of the Petri net model of Fig. 2, there are three routes whose operation places require failure-dependent resources. They are Γ_1, Γ_2 and Γ_3. $\mathcal{N}$ associated with the failure-dependent resources in these routes are:

$\Gamma_1 : \mathcal{N}_1 = \{\mathcal{N}_1^1, \mathcal{N}_1^2\}$; $\Gamma_2 : \mathcal{N}_2 = \{\mathcal{N}_2^1, \mathcal{N}_2^2\}$, and $\Gamma_3 : \mathcal{N}_3 = \{\mathcal{N}_3^7, \mathcal{N}_3^8, \mathcal{N}_3^9\}$. Their respective single-route neighborhoods are: $\mathcal{N}_1^1 = \{p_{14}\}$, $\mathcal{N}_1^2 = \{p_{11}, p_{12}, p_{13}, p_{15}\}$, $\mathcal{N}_2^1 = \{p_{22}, p_{23}, p_{24}, p_{25}, p_{26}\}$, $\mathcal{N}_2^2 = \{p_{21}, p_{27}\}$ $\mathcal{N}_3^7 = \{p_{32}\}$, $\mathcal{N}_3^8 = \{p_{31}\}$, $\mathcal{N}_3^9 = \{p_{33}, p_{34}, p_{35}\}$, with $\mathcal{N}_1^1 \cap \mathcal{N}_1^2 \cap \mathcal{N}_2^1 \cap \mathcal{N}_2^2 \cap \mathcal{N}_3^7 \cap \mathcal{N}_3^8 \cap \mathcal{N}_3^9 = \emptyset$.

Note that our definition of a failure-dependent resource is the same as that of [26]. However, the formulation of single-route neighborhood $\mathcal{N}$ is different from the neighborhood in [26]. We split the neighborhoods according to the parts processing routes that give rise to $\mathcal{N}$. The advantage of breaking up the neighborhoods of failure-dependent resources into their respective routes is to allow constraints formulation based on production lines. This can help us ensure that no markings which do not cause deadlock/blockage, with or without failure events, are prevented from being reached.

We now place restrictions on the number of part type stages that are processed and represented by the operation places, and also resolve conflicts between routes whose operation places share failure-dependent resources. This is necessary if we want to avoid blockage of processing parts that do not require the unreliable resources in the event of unreliable resources failure. Limiting the number of part instances can give a system enough space to move part instances into the buffer spaces of their associated failure-dependent resources until the failed resource is recovered. To do this, we introduce SNC denoted by $\tilde{\mathcal{N}}_j^d$. We construct the constraints in the form of inequality $\tilde{\mathcal{N}}_j^d = M(\mathcal{N}_j^d) \leq B_d$, where $M(\mathcal{N}_j^d)$ is the marking of $\mathcal{N}_j^d$ that denotes the amount of part instances (both finished and unfinished instances) in Γ_j allowed in the neighborhood of the failure-dependent resource r_d, and B_d is the buffer capacity of r_d.

B. SNCs

To derive SNCs our approach is first pathwise. We take one route at time and generate the constraints. However, this will lead to constructing a large number of constraints and some of them may end up being redundant. To account for this issue, we need to reduce the number of constraints by dropping redundant constraints.

Definition 3: Let $\mathcal{N}, \mathcal{N}' \subseteq P_A$. Let $\tilde{\mathcal{N}} = \mathcal{N} \leq b$ and $\tilde{\mathcal{N}}' = \mathcal{N}' \leq b$, with $b \in \mathbb{N}^+$. Constraint $\tilde{\mathcal{N}}$ is said to be redundant if $\tilde{\mathcal{N}} \subseteq \tilde{\mathcal{N}}'$.

Also, an SNC $\tilde{\mathcal{N}}$ constructed from a singleton (1-tuple) set of neighborhood $\mathcal{N}$, i.e., $|\mathcal{N}| = 1$ in a processing route will be considered to be redundant.

Definition 4: Let $\mathcal{P}, \mathcal{P}', \mathcal{P}'' \subseteq P_A$. Constraints $M(\mathcal{P}) < b$, $M(\mathcal{P}') \leq b$, and $M(\mathcal{P}'') < b$ are said to be inseparably implemented in the Petri net model if $\mathcal{P}' \cup \mathcal{P}'' = \mathcal{P}$.

Theorem 1: Inseparable constraints can be implemented in an S^{3}PR class of Petri net model with one control place.

The proof of *Theorem 1* is included in Section III of the supplementary file.

Definition 5: Given a series of failure-dependent resources positioned next to one another $r_{d1}, r_{d2}, \ldots, r_{d(n-1)}, r_{dn}$, in a route Γ_j, such that $r_{d1}, r_{d2}, \ldots, r_{d(n-1)} \in R_f \cap P_R^r$ and $r_{dn} \in R_f \cap (P_R^r \text{ or } P_R^u)$, where $n \geq 2$. Let their respective neighborhoods be $\mathcal{N}_j^{d1}, \mathcal{N}_j^{d2} \ldots, \mathcal{N}_j^{d(n-1)}$, $\mathcal{N}_j^{dn}$ such that $p_{jk} \in \mathcal{N}_j^{d1} : R(p_{jk}) = r_{d1}$, $p_{j(k+1)} \in \mathcal{N}_j^{d2} : R(p_{j(k+1)}) = r_{d2}, \ldots, p_{j(k+n-2)} \in \mathcal{N}_j^{d(n-1)} : R(p_{j(k+n-2)}) = r_{d(n-1)}$, $p_{j(k+n-1)} \in \mathcal{N}_j^{dn} : R(p_{j(k+n-1)}) = r_{dn}$. The set $\mathcal{N}_j^c = \{\mathcal{N}_j^{d1}, \mathcal{N}_j^{d2}, \ldots, \mathcal{N}_j^{d(n-1)}, \mathcal{N}_j^{dn}\}$ is said to be the cluster set of the sets of neighborhood of the series of failure-dependent resources in the route Γ_j if $|\mathcal{N}_j^{d1}| > 1$. Sets of neighborhoods in a set of neighborhoods cluster $\mathcal{N}_j^c$ can be treated as one unit and their buffer capacities can be combined. Let the combined buffer capacities be $B_c = B_{d1} + B_{d2} + \ldots + B_{d(n-1)} + B_{dn}$.

The cluster $\mathcal{N}_j^c$ ends with either $r_i \in R_f$ or $\mathcal{N}_j^i$ belonging to $r_i \in R_f \cap P_R^u$. If $\mathcal{N}_j^{dn}$ belongs to $r_{dn} \in R_f \cap P_R^u$, the cluster $\mathcal{N}_j^c$ is said to be ended by $\mathcal{N}_j^i$ which belongs to an unreliable resource $r_i \in P_R^u$ that is failure-dependent upon itself. On the other hand, if $\mathcal{N}_j^{dn}$ belongs to $r_{dn} \in R_f \cap P_R^r$, the cluster $\mathcal{N}_j^c$ is said to be ended by a non-failure-dependent resource $r_i \in \tilde{\mathrm{R}}_f$. $r_i \in \tilde{\mathrm{R}}_f$ is also a reliable resource $r_i \in P_R^r$.

For convenience, we shall call any constraint that makes another constraint redundant its essential constraint. For the sake of computational and structural simplicity, we shall ignore any redundant constraint since its role in the system can be covered by its essential constraint.

Failure-dependent resources contained in the same resource circuits are also taken into account when deriving the constraints. We compute the set of resource circuits in R_f since we may have some element in R_f routes forming resource circuits.

Definition 6: A directed graph $\mathcal{G}$ of an S^{3}PR Petri net that contains only resource places and transitions is called a resource circuit.

Algorithm 1 in Section IV of the supplementary file describes how to compute a directed graph (digraph) within R_f in a route of an S^{3}PR Petri net model.

Definition 7: Let $r_v, r_w \in R_f$, r_v, r_w are said to be contained in the same resource circuit $\mathcal{C}^{v,w}$ of $\mathcal{G}$ in an S^{3}PR if there is a directed path from r_v to r_w and a directed path from r_w to r_v

Γ_1 is the only route that has resource circuit $\mathcal{C}_1^{1,2} = r_1, t_{14}, r_2, t_{15}, r_1$, and r_1 and r_2 are contained in $\mathcal{C}_1^{1,2}$. It is easy to verify that r_2 is reachable from r_1 through t_{14} and r_1 is reachable from r_2 through t_{15}. We have $p_{14} \in H(r_1)$ with $R(p_{15}) = r_2$ and $p_{13} \in H(r_2)$ with $R(p_{14}) = r_1$. Furthermore, $\{p_{13}, p_{15}\} \subseteq \mathcal{N}_1^2$ and $\{p_{14}\} \subseteq \mathcal{N}_1^1$. As a result, any fair of failure-dependent resources that are contained in the same resource circuit should not be capacitated at the same time. For $r_v, r_w \in R_f$ that are part of a resource circuit $\mathcal{C}_j^{v,w}$ in a route Γ_j, the following constraint must be imposed.

$$\tilde{\mathcal{N}}_j^{v,w} = M(\mathcal{N}_j^v) + M(\mathcal{N}_j^w) < B_w + B_w$$

Computing resource circuits is like computing strongly connected components in a digraph. There are several algorithms for finding the strongly connected components that require only one depth-first search (DFS) traversal in linear time complexity [54], such as the Tarjan Algorithm [55].

1) SNCs IN Γ_1

There are two $\mathcal{N}$s in this route, $\mathcal{N}_1^1$ and $\mathcal{N}_1^2$; therefore we have the following SNCs:

$$\tilde{\mathcal{N}}_1^2 = M(\mathcal{N}_1^2) \leq B_2$$
$$\tilde{\mathcal{N}}_1^1 = M(\mathcal{N}_1^1) \leq B_1$$

We have $\tilde{\mathcal{N}}_1^2$ since $r_2 \in R_f$, $r_2 \in P_R^u$ and the neighborhood $\mathcal{N}_1^2$ is due to $r_2 \in P_R^u$. If r_2 fails while $\tilde{\mathcal{N}}_1^2$ is over-capacitated, there is no buffer space of any related failure-dependent resource in the neighborhood $\mathcal{N}_1^2$ to keep the part type stage, $p_{jk} \in \mathcal{N}_1^2$, temporarily if $M(\mathcal{N}_1^2) > B_2$. This could cause blockage and deadlock. Therefore, any $\mathcal{N}_j^d$ that belongs to $r_i \in P_R^u$ should not be over-capacitated.

We ignore $\tilde{\mathcal{N}}_1^1 = M(\mathcal{N}_1^1) \leq B_1$ since $\mathcal{N}_1^1$ is a singleton $\mathcal{N}$ and therefore, it is a redundant constraint.

As a result of $\mathcal{C}_1^{1,2}$ in the route, there should be a constraint for the pair of neighborhoods of the failure-dependent resources r_1 and r_2 along Γ_1 as follows:

$$\tilde{\mathcal{N}}_1^{1,2} = M(\mathcal{N}_1^1) + M(\mathcal{N}_1^2) < B_1 + B_2$$

Constraint $\tilde{\mathcal{N}}_1^{1,2}$ must hold to guarantee that at most one neighborhood of $\mathcal{N}_1^1$ and $\mathcal{N}_1^2$ is capacitated at a time to prevent deadlocks.

2) SNCs IN Γ_2

This route has $\mathcal{N}_2^1$ and $\mathcal{N}_2^2$. We have $r_1 \in R_f(r_2)$ but $r_1 \in P_R^r$, which makes r_1 not subject to failure and $\mathcal{N}_2^1$ and $\mathcal{N}_2^2$ are not contained in a resource-circuit. Furthermore, $p_{26} \in \mathcal{N}_2^1$ is due to $r_1 \in R_f \cap P_R^r$ and $p_{27} \in \mathcal{N}_2^2$ is due to $r_2 \in R_f \cap P_R^u$. $R(p_{26}) = (r_1)$ and the next part type stage p_{27} requires r_2, i.e., $R(p_{27}) = r_2$. Therefore, $\mathcal{N}_2^1$ and $\mathcal{N}_2^2$ are two adjacent single-route neighborhoods that form a neighborhood cluster $\mathcal{N}_2^{1,2}$. Allowing parts instances up to the capacity of $B_1 + B_2$ in the $\mathcal{N}_2^1$ will not result into blockage or deadlock if r_2 fails. The part instances can all be absorbed by the buffer spaces B_1 and B_2 pending repair/recovery of r_2. Consequently, we have:

$$\tilde{\mathcal{N}}_2^1 = M(\mathcal{N}_2^1) + M(\mathcal{N}_2^2) \leq B_1 + B_2$$
$$\tilde{\mathcal{N}}_2^2 = M(\mathcal{N}_2^2) \leq B_2$$

SNC for the pair of the adjacent neighborhoods, $\mathcal{N}_2^1$ and $\mathcal{N}_2^2$, is as follows:

$$\tilde{\mathcal{N}}_2^{1,2} = M(\mathcal{N}_2^1) + M(\mathcal{N}_2^2) \leq B_1 + B_2$$

$\tilde{\mathcal{N}}_2^{1,2}$ is an essential constraint of both $\tilde{\mathcal{N}}_2^1$ and $\tilde{\mathcal{N}}_2^2$, hence making $\tilde{\mathcal{N}}_2^2$ and $\tilde{\mathcal{N}}_2^1$ redundant. Consequently, the only SNC in Γ_2 that should be considered is $\tilde{\mathcal{N}}_2^{1,2}$.

3) SNCs IN Γ_3

For Γ_3 the following constraints must hold to ensure that no neighborhood of failure-dependent resource is over-capacitated:

$$\tilde{\mathcal{N}}_3^7 = M(\mathcal{N}_3^7) \leq B_7$$
$$\tilde{\mathcal{N}}_3^8 = M(\mathcal{N}_3^8) \leq B_8$$
$$\tilde{\mathcal{N}}_3^9 = M(\mathcal{N}_3^9) \leq B_9$$

However, $\tilde{\mathcal{N}}_3^7$ and $\tilde{\mathcal{N}}_3^8$ are both singletons, and thus redundant when implementing the SNCs in the Petri net model.

4) SNCs FOR ROUTES WITH SHARED FAILURE-DEPENDENT RESOURCES

Definition 8: [56] A resource place $r_i \in P_R$ is called a shared resource place if $|H(r_i)| \geq 2$ and a non-shared one if $|H(r_i)| = 1$.

Let the set of failure-dependent resources shared by multiple routes be S_f. We now consider SNCs due to shared failure-dependent resources and resource circuits as a result of multiple routes. There are only two routes in Fig. 1 whose sets of SNCs share the same failure-dependent resources and mutual flow exist between them. These routes are Γ_1 and Γ_2. $\mathcal{N}_1^1$ and $\mathcal{N}_2^1$ are neighborhoods of r_1 and $\mathcal{N}_1^2$ and $\mathcal{N}_2^2$ are neighborhoods of r_2. To find resource circuits, we construct a digraph by considering only failure-dependent resources and transitions as sets of vertices. To compute resource circuit for multiple paths that have shared failure-dependent resources, first let the union of $\mathcal{N}_1^1$ and $\mathcal{N}_2^1$ be denoted by $\mathcal{N}_{1,2}^1$ and that of $\mathcal{N}_1^2$ and $\mathcal{N}_2^2$ by $\mathcal{N}_{1,2}^2$. Algorithm 2 in Section V of the supplementary file describes how to compute a directed graph within R_f in multiple routes of an S^3PR Petri net model.

It can be verified that there is a path from r_1 to r_2 through t_{14} and a path from r_2 to r_1 through t_{27}. Therefore, the multi-path (Γ_1 and Γ_2) resource circuit that contains r_1 and r_2 is $\mathcal{C}_{1,2}^{1,2} = r_1, t_{14}, r_2, t_{27}, r_1$. Since we have $\{p_{14}, p_{26}\} \subseteq \mathcal{N}_{12}^1$ and $\{p_{13}, p_{27}\} \subseteq \mathcal{N}_{1,2}^2$, we have the following constraint:

$$\tilde{\mathcal{N}}_{1,2}^{1,2} = M(\mathcal{N}_1^1) + M(\mathcal{N}_1^2) + M(\mathcal{N}_2^1) + M(\mathcal{N}_2^2) < B_1 + B_2$$

Given sets of single-route neighborhoods that belong to failure-dependent resources contained in the same resource circuits as a result of their overlapping routes, we must generate a constraint to ensure that the neighborhoods are never capacitated.

Also, $\mathcal{N}_1^2$ and $\mathcal{N}_2^2$ compete for r_2 with $r_2 \in P_R^u$. In order to avoid over-capacitating the neighborhoods of $r_2 \in P_R^u$, the following constraint has to be enforced.

$$\tilde{\mathcal{N}}_{1,2}^2 = M(\mathcal{N}_1^2) + M(\mathcal{N}_2^2) \leq B_1$$

For any sets of single-route neighborhoods that have a shared failure-dependent resource because of their overlapping routes, we must compute a constraint to guarantee that the neighborhoods are never over-capacitated. In summary, these are the constraints that must be imposed on the

TABLE 1. Control places computed for the SNCs.

SNC	Control place	$^\bullet\mathcal{S}$	$\mathcal{S}^\bullet$	$M_0(\mathcal{S})$
$\tilde{\mathcal{N}}^2_{1,2}$	$\mathcal{S}^2_{1,2}$	$t_{14}, t_{16}, t_{22}, t_{28}$	$t_{11}, t_{15}, t_{21}, t_{27}$	1
$\tilde{\mathcal{N}}^9_3$	$\mathcal{S}^9_3$	t_{36}	t_{33}	1
$\tilde{\mathcal{N}}^\star_{1,2}$	$\mathcal{S}^\star_{1,2}$	$2t_{16}, t_{28}$	$2t_{11}, t_{21}$	2

distribution of part types that require unreliable resource in their production process.

$$\begin{aligned}
\tilde{\mathcal{N}}^2_1 &= M(\mathcal{N}^2_1) \leq B_2\\
\tilde{\mathcal{N}}^{1,2}_1 &= M(\mathcal{N}^1_1) + M(\mathcal{N}^2_1) < B_1 + B_2\\
\tilde{\mathcal{N}}^{1,2}_2 &= M(\mathcal{N}^1_2) + M(\mathcal{N}^2_2) \leq B_1 + B_2\\
\tilde{\mathcal{N}}^9_3 &= M(\mathcal{N}^9_3) \leq B_9\\
\tilde{\mathcal{N}}^2_{1,2} &= M(\mathcal{N}^2_1) + M(\mathcal{N}^2_2) \leq B_2\\
\tilde{\mathcal{N}}^{1,2}_{1,2} &= M(\mathcal{N}^1_1) + M(\mathcal{N}^2_1) + M(\mathcal{N}^1_2) + M(\mathcal{N}^2_2) < B_1 + B_2
\end{aligned}$$

We can still get rid of the remaining redundant constraints from the results above. It is obvious that $\tilde{\mathcal{N}}^2_1$ is a redundant constraint due to $\tilde{\mathcal{N}}^2_{1,2}$; likewise $\tilde{\mathcal{N}}^{1,2}_1$ is redundant as a result of $\tilde{\mathcal{N}}^{1,2}_{1,2}$ being its essential constraint. Finally, we have the following set of constraints to implement in the Petri net model:

$$\begin{aligned}
\tilde{\mathcal{N}}^{1,2}_2 &= M(\mathcal{N}^1_2) + M(\mathcal{N}^2_2) \leq B_1 + B_2\\
\tilde{\mathcal{N}}^9_3 &= M(\mathcal{N}^9_3) \leq B_9\\
\tilde{\mathcal{N}}^2_{1,2} &= M(\mathcal{N}^2_1) + M(\mathcal{N}^2_2) \leq B_2\\
\tilde{\mathcal{N}}^{1,2}_{1,2} &= M(\mathcal{N}^1_1) + M(\mathcal{N}^2_1) + M(\mathcal{N}^1_2) + M(\mathcal{N}^2_2) < B_1 + B_2
\end{aligned}$$

By *Definition 4*, $\tilde{\mathcal{N}}^{1,2}_2$, $\tilde{\mathcal{N}}^{1,2}_1$ and $\tilde{\mathcal{N}}^{1,2}_{1,2}$ are inseparable constraints and, by *Theorem 1*, they can be implemented in the Petri net model by using (11) in Section III of the supplementary file.

If we let the inseparable constraints of $\tilde{\mathcal{N}}^{1,2}_1$, $\tilde{\mathcal{N}}^{1,2}_2$ and $\tilde{\mathcal{N}}^{1,2}_{1,2}$ be $\tilde{\mathcal{N}}^\star_{1,2}$, then we have the following three constraints, $\tilde{\mathcal{N}}^\star_{1,2}$, $\tilde{\mathcal{N}}^2_{1,2}$ and $\tilde{\mathcal{N}}^9_3$ to be implemented using control places. The summary of the constraints with their initial markings, presets and postsets transitions is given in Table 1. Algorithm 3 of Section VI of the supplementary file describes the steps for computing the SNCs.

Lemma 1: Given a production route of an S^{3}PR class of Petri net model of a single-unit resource allocation system with a set of unreliable resources P^u_R, SNCs can guarantee that no marking in $\mathcal{M}_{Lu}$ is blocked and all markings in $\mathcal{M}_F$ are disallowed in the neighborhoods of $r_i \in R_f$.

Proof: We use the buffer spaces of the failure-dependent resources to keep part type temporarily before the failed resources are repaired. For any Γ_j, we need to generate maximum of three types of SNCs for the neighborhood of failure-dependent resources: $\tilde{\mathcal{N}}^d_j$, $\tilde{\mathcal{N}}^c_j$ and $\tilde{\mathcal{N}}^{v,w}_j$. We can construct, for any neighborhood, a constraint that allows it to accommodate part instances not exceeding their combined buffers capacities. For any lone and isolated neighborhood, we can generate $\tilde{\mathcal{N}}^d_j \leq B_d$ to allow part instances in the neighborhood of $\mathcal{N}^d_j$, which is its maximum buffer capacity that it can accommodate when there is a failure without causing blockage. Likewise, for any neighborhood cluster $\mathcal{N}^c_j$, we have $\tilde{\mathcal{N}}^c_j \leq B_c$. Any additional part instance to their neighborhood may result in blockage when the unreliable resources with which the neighborhoods are associated fail.

Consider a case in which a part type stage p_{jk} holds a non-failure-dependent resource $r_a \in \tilde{\mathrm{R}}_f \cap P^r_R$, i.e., $R(p_{jk}) = r_a$ and p_{jk+1} requires an unreliable resource $R(p_{jk+1}) = r_b \in R_f \cap P^u_R$. Suppose that r_b is the only unreliable resource in the route Γ_j. This means that $p_{jk}, p_{jk+1} \in \mathcal{N}^b_j$. Suppose that we allow part type stages beyond the capacity of B_b, say $B_b + 1$. If r_b fails, the additional one part type stage will have no buffer space to be kept in temporarily. We have to allow it to stay in the buffer space of r_a, B_a, even after r_a finishes processing it, pending the repair/recovery of r_b. Thus, it could block any part type stage that is not in Γ_j from accessing r_a.

$\tilde{\mathcal{N}}^{v,w}_j$ is for neighborhoods that belong to a pair failure-dependent resources contained in the same resource circuit. The constraints must ensure that $\mathcal{C}$s are not capacitated, since capacitated $\mathcal{C}$s create deadlock. Thus, $\tilde{\mathcal{N}}^{v,w}_j < B_v + B_w$. ■

Lemma 2: Given a number of production routes that have shared $r_d \in R_f$ in an S^{3}PR class of Petri net model of a single-unit resource allocation system, SNCs can guarantee that no marking $M \in \mathcal{M}_{Lu}$ is blocked and all markings in $\mathcal{M}_F$ are disallowed in the neighborhoods of $r_i \in R_f$.

Proof: The same arguments as in Lemma 1 can be made here. For routes that have shared $r_d \in R_f$ we need to generate two SNCs: $\tilde{\mathcal{N}}^d_z$ and $\tilde{\mathcal{N}}^{v,w}_z$. $\tilde{\mathcal{N}}^d_z$ is for $r_d \in R_f$ that is shared by a number of routes. In order to avoid over-capacitating it, we need $\tilde{\mathcal{N}}^d_z \leq B_d$. $\tilde{\mathcal{N}}^{v,w}_z$ is for $r_v, r_w \in R_f$ that are contained in the same resource circuit. The constraints must ensure that $\mathcal{C}$s are not capacitated to avoid deadlocks. Hence, $\tilde{\mathcal{N}}^{v,w}_z < B_v + B_w$. ■

Theorem 2: SNCs are robust supervisory control to failure of P^u_R of an S^{3}PR class of Petri net model of a single-unit resource allocation system.

Proof: To have a robust control for an S^{3}PR of a single-unit resource allocation system with P^u_R, all the neighborhoods of failure-dependent resources must be free of deadlock and blockage, whether there is a failure or not. Additionally, there should be no deadlock or blockage when an unreliable resource is recovered after breaking down. According to Lemmas 1 and 2, for the set of unreliable resources P^u_R, we need to construct at most five types of SNCs to guarantee deadlock/blockage-free operation in the neighborhoods of $r_i \in R_f$. These SNCs are: $\tilde{\mathcal{N}}^d_j$, $\tilde{\mathcal{N}}^c_j$ and $\tilde{\mathcal{N}}^{v,w}_j$ for a single route and $\tilde{\mathcal{N}}^d_z$ and $\tilde{\mathcal{N}}^{v,w}_z$ for routes with shared resources. Since each of these constraints can ensure that no deadlock or blockage occurs within the neighborhood of failure-dependent resources whether there is a resource failure or not and after a failed resource is recovered, we can conclude that SNCs are robust supervisory control to failure of P^u_R of an S^{3}PR class of a single-unit resource allocation system. ■

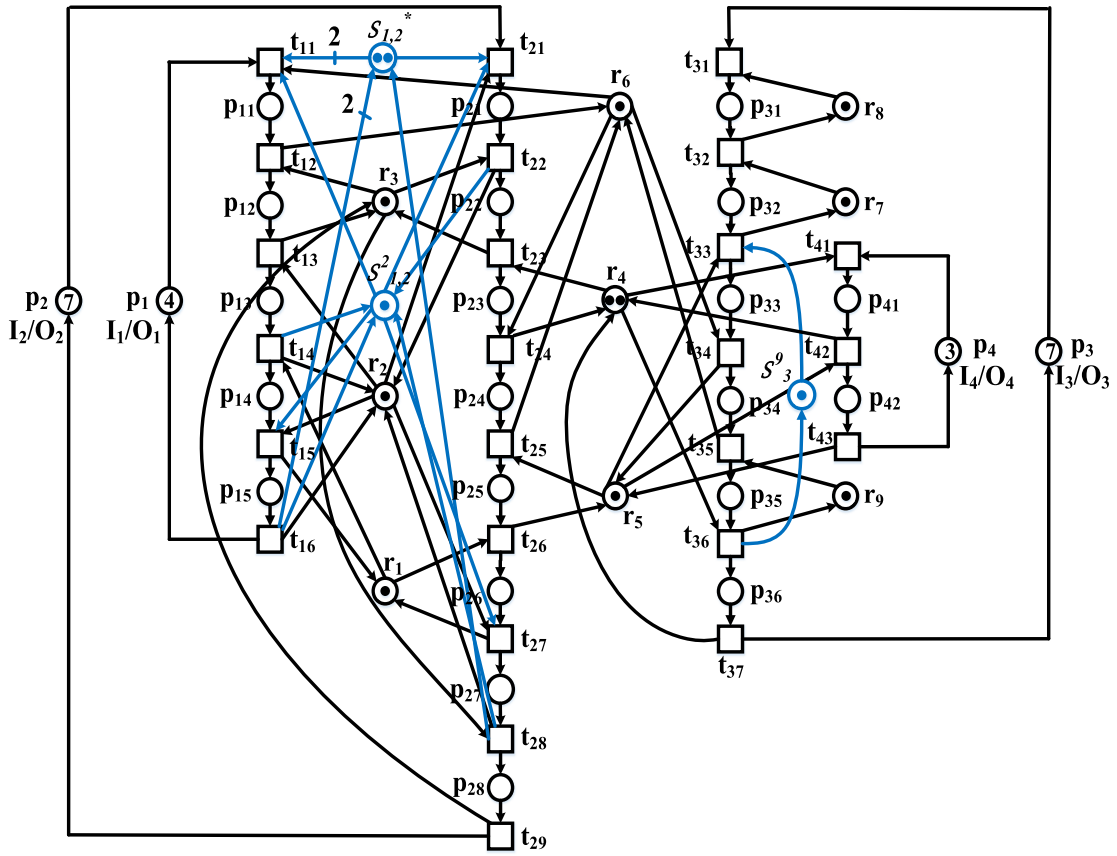


FIGURE 3. Reduced PN model of Fig. 2.

Theorem 3: SNCs are robust supervisory control to failure of P_R^u of an S^3PR class of Petri net model of a single-unit resource allocation system with part types requiring multiple unreliable resources in their processing routes.

Proof: Let $r_v, r_w \in P_R^u, v \neq w$, such that a job type J_m requires both r_v and r_w on its production route Γ_m. Suppose that the job J_m requires r_v first and then r_w. We have $R(p_{mb}) = r_v$ and $R(p_{m(b+c)}) = r_w$, where p_{mb} and $p_{m(b+c)}$ are part type stages of J_m and $b, c \geq 1$. Thanks to $r_v, r_w \in P_R^u$, it holds $r_w, r_v \in R_f$. Thus, we have $\tilde{\mathcal{N}}_m^v = M(\mathcal{N}_m^v) \leq B_v$ and $\tilde{\mathcal{N}}_m^w = M(\mathcal{N}_m^w) \leq B_w$ as the SNCs for r_w and r_v on Γ_m respectively. Therefore, we can use SNCs to handle multiple unreliable resources in a route. Following the same procedure, we can, if there is n unreliable resources in any production route, construct n sets of neighborhoods with their respective constraints to ensure deadlock/blockage-free operations in the route whether there are failures of unreliable resources or not. Thus, SNCs can be used to handle multiple unreliable resources in a route. ■

Section VI of the supplementary file presents Algorithm 3 that describes the steps for computing SNCs and its related theorems and their proofs.

C. SNCs PETRI NET IMPLEMENTATION

Table 1 summarizes the constraints with their initial markings, presets and postsets transitions and Fig. 3 depicts the constrained Petri net model. Let the reduced Petri net model with SNCs be $(N_{\tilde{\mathcal{N}}}, M_{\tilde{\mathcal{N}}0})$. This reduced Petri net model is not deadlock free since SNCs only control the distribution of part instances within failure-dependent resource so that the system is kept free of deadlock and blockage if unreliable resources go down. Therefore, we need a second deadlock control policy to ensure that the remaining parts of the system that do not require the unreliable resources do not enter deadlock states.

D. DESIGNING MONITORS FOR EMPTIABLE SIPHONS

The original buffer net of the plant model (N_u, M_{u0}) in Fig. 1 has 48,108 states, 31,544 of which are in live-zone (LZ) and 16,564 are in deadlock-zone (DZ). More detailed explanations on LZ and DZ could be found in Section II of the supplementary file. The reduced Petri net model of the system $(N_{\tilde{\mathcal{N}}}, M_{\tilde{\mathcal{N}}0})$ has 5,108 states, 4,928 of which are live states and 180 are deadlock states. Since we have established the robustness of SNC in tackling deadlock/blockage within R_f no matter whether an unreliable resource fails or not. Thus, these 180 deadlock states are as result of lack of supervisory control within $\tilde{R}_f$. Therefore, we design monitors in the form of control places to prevent any minimal siphon from being unmarked in the constraint Petri net model $G = (N_{\tilde{\mathcal{N}}}, M_{\tilde{\mathcal{N}}0})$. Basic definitions of siphon and trap can be found in Section I of the supplementary file. Information

about strictly minimum, elementary and dependent siphons can also be found in Section I of the supplementary file.

Property 1 [60]: A Petri net is deadlock-free if for each minimal siphon S, either it contains a marked trap or $F(S) > 0$, where $F(S)$ is a quantity denoted by $F(S) = min\{M(S)|M = M_0 + [N]Y, M, Y \geq 0\}$, where M and Y are real numbers.

Property 2 [60]: $F(S) > 0$ for any siphon that is invariant-controlled.

Theorem 4 [61]: S is controlled if $M_0(S) > \sum_{i=1}^{n} a_i M_0(S_i) - \sum_{i=1}^{n} a_i$, where $a_i \geq 0$.

We adopt the method proposed in [61]. We first apply the MIP-based deadlock detection approach developed in [60] to the constrained Petri net model to find a maximal unmarked siphon S^* that has deadlocks in it. We derive a minimal unmarked siphon S_1 from S^*. We check the dependency of S_1 with respect to other siphons obtained, by either Theorem 4 or $F(S) > 0$ (Property 1). If the controllability of S_1 is guaranteed, we do not add a monitor for it; otherwise we add a monitor V_{s_1} according Proposition 1 in Section I of the supplementary file, and we check for deadlocks again in the plant. We find an unmarked maximal siphon S^* and derive a minimal unmarked siphon S_2 and we repeat the procedure again. Algorithm 4 in Section VII of the supplementary file describes the siphon control method.

The MIP-based in [60] introduces indicators: $v_p = 1\{p \notin S\}$ and $z_t = 1\{t \notin S^\bullet\}$. Obviously, any p with $v_p = 1$ and any t with $z_t = 1$ will be removed from the siphons. Since S is a siphon, $v_p = 0 \Rightarrow z_t = 0, \forall t \in p^\bullet$ and $z_t = 1 \Rightarrow v_p = 1, \forall p \in^\bullet t$, which lead to

$$z_t \geq \sum_{p \in {}^\bullet t} v_p - |{}^\bullet t| + 1, \quad \forall t \in T \tag{1}$$

$$v_p \geq z_t, \quad \forall (t, p) \in F \tag{2}$$

$$v_p, z_t \in \{0, 1\} \tag{3}$$

For a structurally bounded net, we have

$$v_p \geq M(p)/SB(p), \quad \forall p \in P \tag{4}$$

where structurally bound (SB) is defined as $SB(p) = max\{M(S)|M = M_0 + [N]Y, M, Y \geq 0\}$. The following MIP can be used to determine the maximal siphon unmarked at a given marking M if there exist siphons unmarked if $G^{MIP}(M) < |P|$:

$$G^{MIP} = Minimize \sum_{p \in P} v_p$$

subject to constraints (1)–(4) and

$$M = M_0 + [N]Y, \quad M \geq 0, \ Y \geq 0$$

If there is no emptiable siphons at M in a Petri net model $G^{MIP}(M) = |P|$ is true.

If we apply Algorithm 3 to the reduced Petri net model $G = (N_{\tilde{\mathcal{N}}}, M_{\tilde{\mathcal{N}}0})$, we find the maximal unmarked siphon $S^* = \{p_{11}, p_{25}, p_{34}, p_{42}, r_5, r_6\}$. It can be verified that S^* is not a dependent siphon and it is a minimal siphon $S^* = S_1 = \{p_{11}, p_{25}, p_{34}, p_{42}, r_5, r_6\}$. We have $[S_1] = \{p_{24}, p_{33}\}$, $\mathfrak{B}_1^1 = \{p_{21}, p_{22}, p_{23}\}$ and $\mathfrak{B}_1^2 = \{p_{31}, p_{32}\}$. Therefore, $\mathfrak{B}_1 = \mathfrak{B}_1^1 \cup \mathfrak{B}_1^2 = \{p_{21}, p_{22}, p_{23}, p_{31}, p_{32}\}$, and $V_{s_1} + p_{21} + p_{22} + p_{23} + p_{24} + p_{31} + p_{32} + p_{33}$ is a P-invariant with $M_0(V_{s_1}) = M_0(S_1) - 1 = 1$. The set of input and output transitions of V_{s_1} are ${}^\bullet V_{s_1} = \{t_{25}, t_{34}\}$ and $V_{s_1}^\bullet = \{t_{21}, t_{31}\}$, respectively.

If V_{s_1} is added to $(N_{\tilde{\mathcal{N}}}, M_{\tilde{\mathcal{N}}0})$, the resultant net system denoted by $G = (N_c^*, M_c^*)$. We can easily see that $G^{MIP} = 36 = |P|$. This implies that there is no emptiable siphon in G. Thus, the net system is live. The addition of the monitor according Proposition 1 in Section I of the supplementary file. is based on the method proposed in [8]. The method is conservative since it requires that all output arcs of the monitors are added to the source transitions of the original Petri net model (the reduced Petri net model in this case). For example, the resultant net system (N_c^*, M_c^*) has only 1,918 reachable markings if V_{S_1} is added with the output arcs connected to t_{21} and t_{31}. To reduce such restriction, we use an algorithm proposed in [61], which can be found in Section VIII of the supplementary file as Algorithm 5.

First, we remove V_{S_1} including its related arcs, and supposedly add V_{S_1} in such a way that $V_{s_1} + p_{22} + p_{23} + p_{24} + p_{31} + p_{32} + p_{33}$ is a P-invariant. This implies that p_{21} is deleted from $\mathfrak{B}_1$, hence, $\mathfrak{B}_1^{NEW} = \{p_{22}, p_{23}, p_{31}, p_{32}\}$. The resultant net system is denoted by $G = (N_{c1}^*, M_{c1}^*)$. We can easily verify that $G^{MIP} = 36$, implying that the system contains no emptiable siphon. We repeat the same procedure for the elements of $\mathfrak{B}_1^{NEW}$, and each element removed we find that $G^{MIP} = 36$ until $\mathfrak{B}_1^{NEW} = \emptyset$. This implies that the final P-invariant obtained $V_{s_1} + p_{24} + p_{33}$ can prevent S_1 from being emptied, since there is no more emptiable siphon in the final net system denoted by $G = (N_c, M_c)$. The set of input and output transitions of V_{s_1} are ${}^\bullet V_{s_1} = \{t_{25}, t_{34}\}$ and $V_{s_1}^\bullet = \{t_{24}, t_{33}\}$, respectively. The final controlled Petri net model obtained (N_c, M_c) has 4,928 reachable live markings, implying that all the live states in the constrained Petri net model are reachable.

In one of our assumptions in Section 2, we state that an unreliable r_u resource can only fail while it is operating on a part type stage p_{jk}. If the failure of the unreliable resource occurs before it finishes processing the part type stage, the part type stage remains in the buffer space of the unreliable resource until the resource is recovered. After its recovery, the unreliable resource finishes processing the part type stage before the part type stage moves to the next resource or workstation, or moves out of the system if it is at its last processing stage. In the Petri net model, this situation can be modelled by disenabling the output transition $t_{j,k+1}$ of p_{jk}. Since disenabling $t_{j,k+1}$ prevents it from firing and releasing tokens into $p_{j,k+1}$ and r_u. Hence, token(s) in $p_{j,k}$ remain there until the $t_{j,k+1}$ is enabled, which means recovery of r_u. This means that $t_{j,k}$ becomes a dead transition when r_u fails, which implies a deadlock in the system. Since

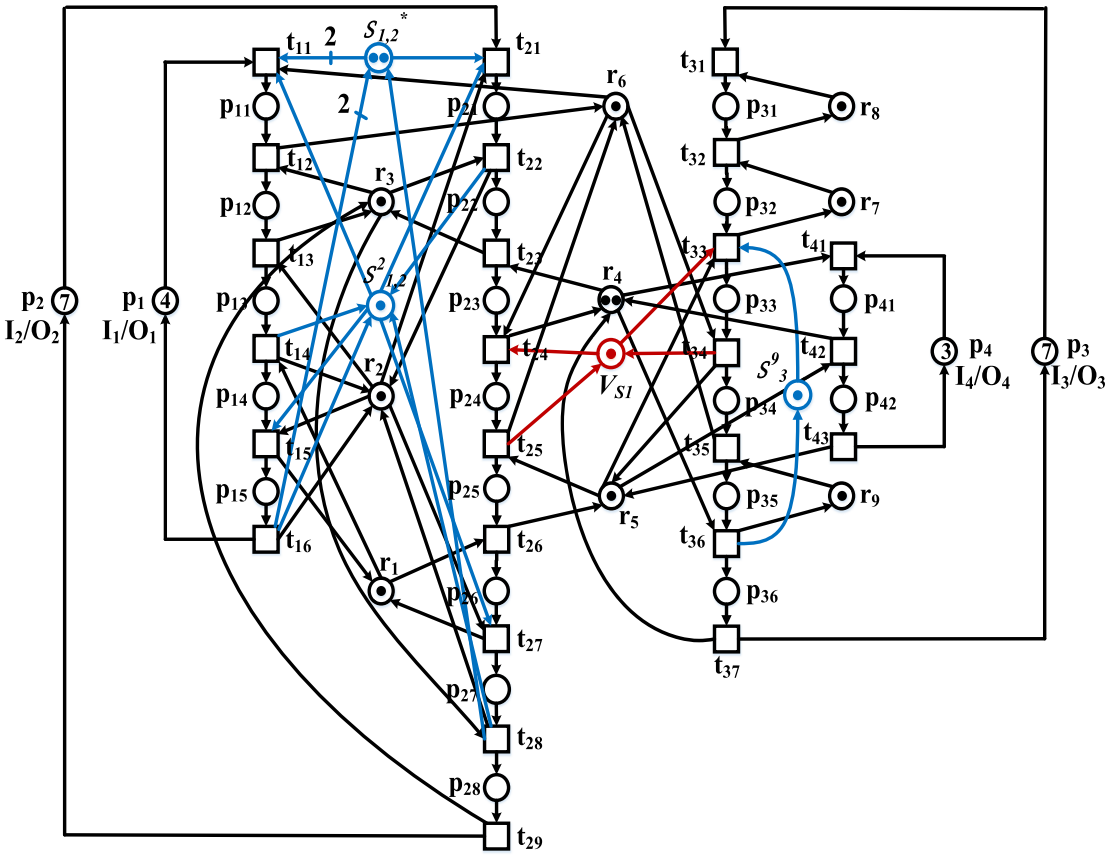


FIGURE 4. Controlled Petri net model.

we have a robust supervisory controller in the system, this deadlock condition does not affect the other parts of the system that do not require r_u. Any part type stage that requires the failed unreliable resource r_u later in its processing stage will be stored in a buffer space of resources failure dependent on r_u to prevent blockage in the other parts of the system.

Table 2 shows the behaviors of the Petri net of the system in Fig. 1 under different unreliable resources conditions. We carried out the simulation using integrated net analyzer (INA), assuming that the set of transitions $T_c = \{t | t \in p^{\bullet}_{j,k}, p_{j,k} \in H(r_u)\}$ is a set of controllable transitions. Therefore, any $t \in T_c$ is a controllable transition and $T_u = T \setminus T_c$ is set of uncontrollable transitions.

Theorem 3 proves the robustness of SNC and its correctness. The correctness of the second method (let it be Ξ_{sip}) of designing controllers for siphons that can be unmarked at a given marking M has been established in the literature [15], [60], [61], [63], [64], some of which can be found in Section II of the supplementary file of this paper. This leads us to the correctness of our proposed policy. Let the policy that combines SNC and Ξ_{sip} be Ψ. Therefore, $\Psi = \text{SNC} \wedge \Xi_{sip}$ can guarantee deadlock/blockage-free operation in an S^3PR Petri net model of an AMS with unreliable resources of the type considered in this work.

TABLE 2. Number of markings reachable under different unreliable resources conditions.

Condition of unreliable resource	Dead transition	No. reachable markings
There is no resource failure	No dead transition	4,928
r_2 fails while operating on p_{13}	t_{14}	4,608
r_2 fails while operating on p_{15}	t_{16}	4,928
r_2 fails while operating on p_{21}	t_{22}	1,076
r_2 fails while operating on p_{35}	t_{28}	3,580
r_9 fails while operating on p_{35}	t_{36}	2,712

V. COMPARISON WITH PREVIOUS STUDIES

Note that the work in [26] is based on the automata and it is a deadlock avoidance policy implemented online. Both NHCs and Banker's algorithm or SSL are implemented online in [26]. In this work, we implement SNCs and control for emptiable siphon offline in the Petri net model. Both NHCs and SNCs are used for controlling flow of part types that require the services of unreliable resources within their respective failure-dependent resources. In light of these, our bases for comparison between these two policies will be the neighborhood policy and single-route neighborhood policy. Since NHCs are also constraints used to restrict the flow of part types within failure-dependent resources, they can also be implemented in the Petri net framework and the resulting Petri net model will be a unified one. Implementing NHCs in the Petri net model off-line requires connecting control places and their related arcs to the operation places that require

TABLE 3. Control places computed for the NHCs.

NHC	Z_d^i	$^\bullet$NHC	NHC$^\bullet$	M_0(NHC)
NHC_1^2	Z_1^2	t_{15}, t_{27}	t_{14}, t_{22},	1
	Z_2^2	$t_{14}, t_{16}, t_{22}, t_{28}$	$t_{11}, t_{15}, t_{21}, t_{27}$	1
NHC_2^2	$Z_1^2 + Z_2^2$	t_{16}, t_{28}	t_{11}, t_{21}	1
NHC_1^9	Z_7^9	t_{33}	t_{32}	1
	Z_8^9	t_{32}	t_{31}	1
	Z_9^9	t_{36}	t_{33}	1

failure-dependent resources according to the constraints. For the purpose of comparison, Table 3 depicts how NHCs are implemented offline in the Petri net model of Fig. 2.

We have verified that if NHCs are used in the system and implemented in the Petri net model using control places, the resulting number of reachable states will be 2,234. Among them, 24 are deadlock states while 2,208 are live states. Therefore, the maximum number of legal states is 2,208. This shows that SNC policy with 5,108 states is more permissive than NHC policy. SNC policy allows processing of only one part type stage of J_1 at a time, two part type stages of J_2 at a time, three part type stages of J_3, and J_1 and part type stages of J_2 cannot be processed concurrently. On the other hand, NHC policy processes only a part type stage of either J_1 or J_2 at a time.

In terms of computational complexity, both NHC policy and SNC policy are of polynomial time. However, with large-scale systems the computational overhead of the neighborhood constraints may increase significantly. This may also increase the runtime cost, particularly if implemented in online mode due to memory requirement. Offline implementation might be a good idea to tackle possible increase in the computational complexity, if the offline computational cost is not a critical issue, the indicators to evaluate the control strategy are the generality of considered systems and the behavioral permissiveness of the resulting controlled systems. On the other hand, if we consider structural complexity, SNC policy requires less control places than NHC one as it requires less constraints, especially using Petri net models because of their superior structural properties that allow us to drop redundant constraints compared to an automaton. The concept of inseparable constraints for SNC policy gives us another advantage to derive less constraints. For example, consider the number of control places required to add to the Petri net model to implement SNCs. We need to add only three control places to the Petri net model, while implementing NHC policy requires four control places. Additionally, inseparable constraints and clustering of neighborhoods allow us to achieve more admissible number of states in neighborhoods that have mutual flow. Table 4 summarizes the comparison between SNC policy and NHC policy.

The fundamental difference between SNC policy in this study and NHC policy [26] is the number of part types allowed in neighborhoods that form strongly connected components or failure-dependent resources that are contained the same resource transition circuits. For instance, consider

TABLE 4. Comparison between NHC policy and SNC policy as implemented in the Petri net model.

Supervisor	NHC	SNC
No. control places	4	3
No. arcs	18	14
No. reachable markings without resource failure	2,234	5,108

the AMS in Fig. 1. NHCs do not allow markings such as $M = 4p_1 + 5p_2 + 7p_3 + 4p_4 + p_{26} + p_{27} + r_3 + 2r_4 + r_5 + 2r_6 + r_7 + r_8 + r_9$, which is admissible and a legal marking under SNC policy. In fact, NHC policy suppresses any marking that leads to markings at which both p_{26} and p_{21} are marked or both p_{26} and p_{27} are marked at the same time. As a result of this suppression, a significant number of markings are disallowed, which do not result in any deadlock or create blocking effects under failure conditions or lack thereof. Moreover, NHCs alone, unlike SNCs, cannot be used for systems in which part types require multiple unreliable resources in their processing routes. Central buffers have to be used in [32] in order to apply NHCs to handle multiple unreliable resources in a production route of an AMS. However, SNCs can be used directly, without using central buffers, to tackle multiple unreliable resources failure in one route.

The study in [43] combines automata and Petri nets to model, analyze and develop a robust supervisory control for AMSs using a buffer net constraint and generalized neighborhood constraints. The buffer net constraint is constructed using Petri net model. The generalized neighborhood constraint policy is similar to NHC policy in [26] and SNC policy. However, like NHC policy, the generalized neighborhood constraint is established in the framework of automata as an online deadlock avoidance policy.

The studies in [27], [38]–[42], [45], [46], [48]–[50] utilize Petri nets formalism to develop robust supervisory control policies for AMSs. The works in [48] and [49] use DES and Petri net model to design robust control for AMSs deadlock avoidance policies. The policy in [48] is based on a modified banker's algorithm and the method is more permissive than the methods in [26] and [31]. However, the policy in [48] is for a class of AMSs with a single unreliable resource. The method in [49] requires additional buffers as special type of resources to keep part type temporarily to avoid blockage if an unreliable resource fails. This increases the structural complexity of the supervisor.

The policies in [27], [38]–[42], [45], [50] are robust deadlock prevention methods using Petri nets. These policies require addition of recovery subnets to the Petri net models of AMSs with unreliable resources to model resource failures and recoveries. Robust supervisory control is achieved by adding control places to the original Petri net models of AMSs with unreliable resources. However, the works in [27], [38]–[42], [45], [50] are fundamentally different from this work. First, the types of AMSs considered in this work are different from the ones considered in [27], [38]–[42], [45], [50]. In [27], [38]–[42], [45], [46], [50] a resource type

TABLE 5. Comparison of robust control policies based on PN.

Policy	Type of deadlock policy	Can the policy handle AMS with multiple unreliable resources?	Type of resources	Recovery subnet
[27]	Prevention	Yes	Machine/robot	Yes
[38]	Prevention	Yes	Machine/robot	Yes
[39]	Prevention	Yes	Machine/robot	Yes
[40]	Prevention	Yes	Machine/robot	Yes
[41]	Prevention	Yes	Machine/robot	Yes
[42]	Prevention	No	Workstation with buffer slots and server	No
[45]	Prevention	No	Machine/robot	Yes
[46]	Prevention	No	Machine with buffer slots/robot	Yes
[48]	Avoidance	No	Workstation with buffer slots and server	No
[49]	Avoidance	Yes	Workstation with buffer slots and server	No
[50]	Prevention	No	Machine/robot	Yes
This paper	Prevention	Yes	Workstation with buffer slots and server	No

consists of robots and machines without buffer spaces, while in this work a resource type is a workstation composed of buffer spaces that accommodate and hold part types, and a server or processor that processes parts occupying the buffer spaces. Second, a resource failure in this work means the failure of a processor or any piece of equipment in the workstation that makes processing of parts impossible. However, the buffer space of the workstation can still be used since the failure of the workstation does not affect its buffer space. As a result of these differences, the idea of robustness and robust control objectives in these studies are completely different from those in this paper. For robust deadlock control in [27], [38]–[42], [45], [46], [50], there must exist more than one type of an unreliable resource, while in this study, failure of an unreliable does not affect the processing of part types that do not require the failed resource. To avoid repetition here, readers can refer to [51] for more details about the performance analysis and comparison of these robust deadlock control policies [2], [26]–[37], [40], [42], [43], [45], [48]–[50] and other policies not mentioned in this study. Table 5 compares these robust supervisory control policies based on Petri net model. The second column reveals the type of deadlock control strategy used in the related research. The third column shows the type of resource used in developing a policy. The fourth indicates whether a policy can be applied for AMS with multiple unreliable resources, while the fifth reports whether a recovery subnet is used in a policy.

This study is limited to only an S^3PR class of Petri net model due to the type of AMS and the type resource considered. The AMS considered is a single unit resource allocation system where a part type requires a single unit of a single resource type. The Petri net model of the system represents only the buffer spaces and the movements among them. Therefore, all resources or workstations are buffer spaces. As a result, resource allocation and deallocation mean allocation and deallocation of buffer spaces. In practice, it is highly unlikely to have a situation whereby a part type instance (a work piece) requires more than one buffer space of a workstation. The Petri net model of such a system and its resource allocation and deallocation fit only the definition of an S^3PR class of Petri net model.

VI. CONCLUSION

In this study a Petri net-based supervisory control policy that can tackle both deadlock and blockage in automated manufacturing systems with unreliable resources is presented. The policy can keep an AMS away from deadlock both absence or presence of a resource. It has been demonstrated that the proposed policy has higher permissiveness than the original neighborhood policy. Future works should consider dealing with failures, their analysis and recovery. We can look at a system's conditions (behaviors) under failure of its certain parts, and how we can handle recovery process while ensuring continued operations of the unaffected parts. There is also a need for more studies of robust supervisory control for more complex systems such as systems that require multiple units of resources and multiple resources at the same time and flexible routing as well as uncontrollable and unobservable events in AMSs to investigate robust supervisory control issues. Future work also includes applying the proposed methods to social networks and automated manufacturing systems modeled with automata [65], [66].

REFERENCES

[1] N. Wu and M. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 17, no. 5, pp. 658–669, Oct. 2001.
[2] M. A. 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.
[3] K. Barkaoui and J. F. Pradat-Peyre, *On Liveness and Controlled Siphons in Petri Nets* (Lecture Notes in Computer Science), vol. 1091. Berlin, Germany: Springer, 1996, pp. 57–72.
[4] F. Basile, P. Chiacchio, and A. Giua, "An optimization approach to Petri net monitor design," *IEEE Trans. Autom. Control*, vol. 52, no. 2, pp. 306–311, Feb. 2007.
[5] H. Chen, N. Wu, Z. Li, and T. Qu, "On a maximally permissive deadlock prevention policy for automated manufacturing systems by using resource-oriented Petri nets," *ISA Trans.*, vol. 89, pp. 67–76, Jun. 2019.
[6] Y. F. Chen, Z. W. Li, K. Barkaoui, N. Q. Wu, and M. C. Zhou, "Compact supervisory control of discrete event systems by Petri nets with data inhibitor arcs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 365–379, Mar. 2017.

[7] 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.

[8] J. Ezpeleta, J. M. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.

[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] C. Gu, Z. W. Li, N. Q. Wu, M. Khalgui, T. Qu, and A. Al-Ahmari, "Optimization of deterministic timed weighted marked graphs," *IEEE Access*, vol. 6, pp. 68824–68838, 2018.

[11] Z. He, Z. W. Li, and A. Giua, "Improved multi-step look-ahead control policies for automated manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1084–1095, Oct. 2017.

[12] Y. Huang, Y. Pan, and M. Zhou, "Computationally improved optimal deadlock control policy for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 2, pp. 404–415, Mar. 2012.

[13] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer, 2009.

[14] Z. Li, G. Liu, M. Hanisch, and M. Zhou, "Deadlock prevention based on structure reuse of Petri net supervisors for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 1, pp. 178–191, Jan. 2012.

[15] Z. W. Li and M. C. Zhou, "Control of elementary and dependent siphons in Petri nets and their application," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 133–148, Dec. 2008.

[16] Z. W. Li, M. C. Zhou, and M. D. Jeng, "A maximally permissive deadlock prevention policy for FMS based on Petri net siphon control and the theory of regions," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 1, pp. 182–188, Jan. 2008.

[17] 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.

[18] G. Y. Liu, D. Y. Chao, and F. Yu, "Control policy for a subclass of Petri nets without reachability analysis," *IET Control Theory Appl.*, vol. 7, no. 8, pp. 1131–1141, 2013.

[19] G. Y. Liu and K. Barkaoui, "Control policy for a subclass of Petri nets without reachability analysis," *IET Control Theory Appl.*, vol. 46, no. 7, pp. 1147–1160, 2015.

[20] G. Y. Liu and K. Barkaoui, "A survey of siphons in Petri nets," *Inf. Sci.*, vol. 363, pp. 198–220, Oct. 2016.

[21] 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.

[22] L. Piroddi, R. Cordone, and I. Fumagalli, "Selective siphon control for deadlock prevention in Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1337–1348, Nov. 2008.

[23] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A structural approach to the problem of deadlock prevention in processes with resources," in *Proc. Int. Workshop Discrete Event Syst.*, Cagliari, Italy, 1998, pp. 273–278.

[24] M. Uzam, Z. W. Li, and U. S. Abubakar, "Think globally act locally approach for the synthesis of a liveness-enforcing supervisor of FMSs based on Petri nets," *Int. J. Prod. Res.*, vol. 54, no. 15, pp. 4634–4657, 2016.

[25] C. Zhong, W. He, Z. Li, N. Wu, and T. Qu, "Deadlock analysis and control using Petri net decomposition techniques," *Inf. Sci.*, vol. 482, pp. 440–456, May 2019.

[26] S. F. Chew and M. A. Lawley, "Robust supervisory control for production systems with multiple resource failures," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 3, pp. 309–323, Jul. 2006.

[27] G. Y. Liu, Z. W. 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.

[28] 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, 2014.

[29] S. Y. Wang, S. F. Chew, and M. A. 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, Apr. 2008.

[30] S. Foh Chew, S. Wang, and M. A. 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.

[31] H. Yue and K. Y. Xing, "Robust supervisory control for avoiding deadlocks in automated manufacturing systems with one specified unreliable resource," *Trans. Inst. Meas. Control*, vol. 36, no. 4, pp. 435–444, 2014.

[32] S. F. Chew, S. Y. Wang, and M. A. Lawley, "International journal of computer integrated manufacturing," *Trans. Inst. Meas. Control*, vol. 24, no. 3, pp. 229–241, 2011.

[33] H. Yue, K. Y. Xing, H. S. Hu, W. M. Wu, and H. Y. Su, "Robust supervision using shared-buffers in automated manufacturing systems with unreliable resources," *Comput. Ind. Eng.*, vol. 83, pp. 139–150, May 2015.

[34] S. Wang, S. F. Chew, and M. Lawley, "Guidelines for implementing robust supervisors in flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 47, no. 23, pp. 6499–6524, Dec. 2008.

[35] F.-S. Hsieh, "Robustness of deadlock avoidance algorithms for sequential processes," *Automatica*, vol. 39, no. 10, pp. 1695–1706, 2003.

[36] F. S. Hsieh, "Analysis of flexible assembly processes based on structural decomposition of Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 5, pp. 792–803, Sep. 2007.

[37] F.-S. Hsieh, "Robustness analysis of non-ordinary Petri nets for flexible assembly systems," *Int. J. Control*, vol. 83, no. 5, pp. 928–939, May 2010.

[38] G. Liu, P. Li, N. Wu, and L. Yin, "Two-step approach to robust deadlock control in automated manufacturing systems with multiple resource failures," *J. Chin. Inst. Eng.*, vol. 41, no. 4, pp. 484–494, Oct. 2018.

[39] G. Liu, L. Zhang, Y. Liu, Y. Chen, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems based on the max-controllability of siphons," *IEEE Access*, vol. 7, pp. 88579–88591, 2019.

[40] G. Y. Liu, P. Li, Z. W. Li, and N. Q. 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.

[41] G. Y. Liu, L. C. Zhang, L. Chang, A. Al-Ahmari, and N. Q. Wu, "Robust deadlock control for automated manufacturing systems based on elementary siphon theory," *Inf. Sci.*, vol. 510, pp. 165–182, Feb. 2020.

[42] 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.

[43] H. Yue, K. Y. Xing, H. S. Hu, W. M. Wu, and H. Y. Su, "Resource failure and buffer space allocation control for automated manufacturing systems," *Inf. Sci.*, vol. 450, pp. 392–408, Jun. 2018.

[44] J. Luo, Z. Liu, S. Wang, and K. Xing, "Robust deadlock avoidance policy for automated manufacturing system with multiple unreliable resources," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 3, pp. 812–821, May 2020.

[45] 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.

[46] Y. Feng, K. Xing, M. Zhou, X. Wang, and H. Liu, "Robust deadlock prevention for automated manufacturing systems with unreliable resources by using general Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3515–3527, Oct. 2020.

[47] X. Y. Li, G. Y. Liu, Z. W. Li, N. Q. Wu, and K. Barkaoui, "Elementary siphon-based robust control for automated manufacturing systems with multiple unreliable resources," *IEEE Access*, vol. 7, no. 1, pp. 21006–21019, 2019.

[48] J. C. Luo, K. Y. Xing, and Y. C. Wu, "Robust supervisory control policy for automated manufacturing systems with a single unreliable resource," *Trans. Inst. Meas. Control*, vol. 39, no. 6, pp. 793–806, 2017.

[49] X. Wang and H. Hu, "A robust control approach to automated manufacturing systems allowing multitype and multiquantity of resources with Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3499–3514, Oct. 2020.

[50] Y. C. Wu, K. Y. Xing, J. C. Luo, and Y. X. Feng, "Robust deadlock control for automated manufacturing systems with an unreliable resource," *Inf. Sci.*, vol. 346, pp. 17–28, Jun. 2016.

[51] N. Du, H. S. Hu, and M. C. Zhou, "A survey on robust supervisory control policies for automated manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 346, no. 1, pp. 389–406, Sep. 2020.

[52] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, 1996.
[53] 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.
[54] M. H. Alsuwaiyel, *Algorithms: Design Techniques and Analysis*. Singapore: World Scientific, 2000.
[55] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
[56] D. Liu, Z. W. Li, and M. C. Zhou, "Liveness of an extended S^3PR," *Automatica*, vol. 46, pp. 1008–1018, Jun. 2010.
[57] M. Uzam and M. Zhou, "An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 44, no. 10, pp. 1987–2030, 2006.
[58] K. Xing, M. C. Zhou, F. Wang, H. Liu, and F. Tian, "Resource-transition circuits and siphons for deadlock control of automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 1, pp. 74–84, Jan. 2011.
[59] Z. Li, S. Zhu, and M. Zhou, "A divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 2, pp. 156–169, Mar. 2009.
[60] F. Chu and X.-L. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 793–804, Dec. 1997.
[61] Z. W. Li and M. C. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 313–325, Nov. 2006.
[62] M. Fanti, B. Maione, and B. Turchiano, "Studies in informatics and control," *SIAM J. Comput.*, vol. 7, no. 4, pp. 343–364, 1998.
[63] Y. Huang, M. Jeng, X. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 2, pp. 283–305, 2001.
[64] Z. W. Li, H. S. Hu, and A. R. Wang, "Design of liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 4, pp. 517–526, Jul. 2007.
[65] L. Yang, Z. H. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Q. Wu, "On multiplexity-aware influence spread in social networks," *IEEE Access*, vol. 8, pp. 106705–106713, 2020.
[66] S. A. Zhou, Z. H. Yu, E. S. A. Nasr, H. A. Mahmoud, E. M. Awwad, and N. Q. Wu, "Homomorphic encryption of supervisory control systems using automata," *IEEE Access*, vol. 8, pp. 147185–147198, 2020.

UMAR SULEIMAN ABUBAKAR received the B.Eng. degree in electrical engineering from Bayero University, Kano, Nigeria, in 2008, and the M.Sc. degree in electrical and computer engineering from Meliksah University, Kayseri, Turkey, in 2014. He is currently pursuing the Ph.D. degree in control theory and control engineering with the School of Electro-Mechanical Engineering, Xidian University, Xi'an, China. His research interests include Petri net theory and applications and robust supervisory control of automated manufacturing systems.

GAIYUN LIU (Senior Member, IEEE) received the B.S. degree in measurement and control technology and instrumentation and the Ph.D. degree in mechanical engineering from Xidian University, Xi'an, China, in 2006 and 2011, respectively. In 2011, she joined Xidian University, where she is currently an Associate Professor with the School of Electro-Mechanical Engineering. She has authored or coauthored 27 publications. Her research interests include Petri net theory and applications, supervisory control of discrete event systems, and robust supervisory control of automated manufacturing systems.

MURAT UZAM received the B.S. and M.Sc. degrees from the Electrical Engineering Department, Yıldız Technical University, Istanbul, Turkey, in 1989 and 1991, respectively, and the Ph.D. degree from the University of Salford, Salford, U.K., in 1998. From 1993 to 2010, he was with the Department of Electrical and Electronics Engineering, Niğde University, Turkey, as a Research Assistant, an Assistant Professor, an Associate Professor, and a Professor. From 2011 to 2016, he was a Professor with the Department of Electrical and Electronics Engineering, Melikşah University, Kayseri, Turkey. Since April 2020, he has been a Professor with the Department of Electrical and Electronics Engineering, Yozgat Bozok University, Yozgat, Turkey. He has published 47 conference papers and 106 journal and magazine articles, 70 of which are indexed by Science Citation Index Expanded (SCIE). He has published two books in Turkish and four books in English by CRC Press (Taylor and Francis Group). His current research interests include design and implementation of discrete event control systems modelled by Petri nets and, in particular, deadlock prevention/liveness enforcing in flexible manufacturing systems, programmable logic controllers (PLCs), microcontrollers (especially PIC microcontrollers), and design of microcontroller-based PLCs.