Received August 2, 2020, accepted August 30, 2020, date of publication September 8, 2020, date of current version September 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022694

Fault-Recovery and Repair Modeling of Discrete Event Systems Using Petri Nets

EBRAHIM ALI ALZALAB[1], ZHENHUA YU[2], (Member, IEEE), NAIQI WU[3], (Fellow, IEEE), AND HUSAM KAID[4]

[1]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[2]Institute of Systems Security and Control, College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
[3]Institute of Systems Engineering and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Taipa 999078, Macao
[4]Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia

Corresponding author: Zhenhua Yu (zhenhua_yu@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61873277, and in part by the Key Research and Development Program of Shaanxi Province under Grant 2019GY-056.

ABSTRACT Despite advances in automated manufacturing systems (AMSs), faults occur from time to time, which cannot be avoided in a complex real system. A fault is one of the primary causes of failures making some AMS operations unable to complete, and the diagnosis is one of the most important steps in fault-recovery and repair. This work develops a methodology for investigating the behavior of faults on the resources in discrete event systems that are failure-prone. We tackle the fault-tolerant problem and propose a method to make the system able to continue performing its duties, while the failed resources are under a repair and recovery process. In this work, a failure-safe model is proposed and, at the same time, a method for fault recovery and repair of a faulty element is presented without interrupting task processing due to the occurrence of fault to some elements. We use redundant elements to replace the target elements, and these redundant elements are used to do the same work as the target elements do, when faults occur to the target elements. A target element is an unreliable element that is prone to failure. After a faulty target element is repaired and recovered, its failure model is automatically replaced by its repaired model to indicate that the corresponding element has returned to work. The proposed method is tested using an application example. The results show, compared with those obtained by the studies in the literature, that the proposed method has a great performance and outperforms the existing studies.

INDEX TERMS Discrete event system, fault-recovery and repair, fault-tolerance, supervisory control, Petri net.

I. INTRODUCTION

Many contemporary applications in communication networks, transportation, synthesis, and modeling of industrial processes are characterized as dynamic evolution of discrete events. Systems that evolve in a discrete event manner are known as discrete event systems. Typically, systems modeled by a finite state automaton are such an example.

There are several methods for controller design of discrete event systems depending on the models used. However, most of these methods often involve comprehensive system behavior simulations or searches, making them impractical due to the large number of states and events of such systems.

Owing to their modeling capacity and mathematical properties, Petri nets are a very useful and effective tool for studying discrete event systems [6], [7], [20]. They are extensively used in deadlock analysis and control [30], [34], [38]–[42], scheduling [25], [59]–[62], supervisory control [26], [27], [29], [31], [35], [36]. In addition, Petri nets are beneficial to design and analysis of discrete event systems or other formal systems [32], [33], [44]–[56] since they have a simple mathematical representation using a linear matrix of algebra [2], [7]. Also, the modeling paradigm can be used in social networks [22], [23], multi-agent systems [21], [24] and data analysis systems [37].

Fault-tolerance, fault-recovery, and fault-repair in a system are important issues that should be taken into account in many industrial processes [19], [28], [19], [49]–[51], [55].

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

When a fault happens, the goal of a fault-tolerant controller is to diagnose and then repair it such that the system is recovered to its normal state. Thanks to the model simplicity with an intuitive graphical representation, Petri nets have the advantage over other modeling and simulation techniques, which enables a simple visualization of fault-recovery and repair systems [16], [51], [55], [57]. Petri nets can represent fault-recovery and repair systems in a top-down form at different levels of abstraction and detail, and they have a well-developed mathematical foundation that enables both quantitative and qualitative analysis of such systems [56].

Supervisory controllers are usually reconfigured or redesigned to make a system continue its operations after the detection of a fault. In [3], the authors introduce a framework for fault-tolerant monitoring of discrete event systems and present two specifications for non-faulty operations of the overall plant. In [4], a method of designing the controller as an independent supervisory system instead of embedding a controller into the main supervisory system is proposed. Thus, if there is a fault, the controller can be switched to a new controller designed for this particular case. Ioradache and Antsaklis [13] present a method to rearrange the model. Knowledge-based models and neural networks are used by Miyagi and Riascos [14] to deal with each fault.

It is of great importance to maintain the integrity of a system in controller design, fault recovery and fault repair. It requires the knowledge of a system and its elements to keep the system safe after a fault occurs [10]. A fault can occur due to various causes such as internal system incidents, changes in environmental conditions, incorrect operation control behavior, and system design errors [5]. Nowadays, fault tolerance is considered in many manufacturing processes as faults are almost inevitable in automated systems [1]. The types of faults that may occur in systems vary in terms of their continuity: either persistent or intermittent [11]. A fault-tolerant controller allows a system to continue its operations in its degrading specifications with intermittent faults, while it is difficult to do so with persistent faults (faults that prevent the system from performing a task).

In this article, we tackle the fault-tolerant problem in another way and propose a method to make the system able to continue performing its duties, while acts are taken to repair and recover the fault. A method is introduced by Nazemzadeh *et al.* [9] to deal with some of the most important issues in designing a fault-tolerant controller to preserve the safety of a system. Petri nets are used to model such processes with places representing the status of target elements that are failure-prone. To make a system fault-tolerant, there should be a redundant element for each target element. When a fault occurs to a target element, the sub-model of its redundant element is used to replace the model of the faulty target element.

The work in Zhou and DiCesare [56] present four adaption methods for error recovery: input conditioning, alternate path, backward error recovery, and forward error recovery. These methods are used to augment the Petri net controller to ensure some essential properties of the controller, i.e., liveness, boundedness, and reversibility. The fault detection is performed using watchdog timers that indicate a fault when a token stays longer than a certain time in a place. We observe that the resulting controlled system can recognize certain faults, but the system performance during faults may remain to be a degraded one since the system cannot carry out some tasks while the failed resources are being repaired and recovered. Liu *et al.* [16] propose a three-step robust deadlock control strategy for systems with unreliable elements. The proposed strategy designs recovery subnets and monitors for some resource failures in the Petri net model. Normal and inhibitor arcs are inserted to connect the recovery subnets with monitors. We observe that the main weakness of this proposed strategy is that many recovery subnets with inhibitor arcs are added for unreliable resources, which leads to high structural complexity of the initial model. In addition, the proposed method does not guarantee that the system is able to continue performing its duties, while the failed resources are under their repair and recovery process. In addition, Li *et al.* [57] propose elementary siphon-based robust controllers that can handle the multi-type and multi-unit of resource failures. We observe that the proposed approach is not maximally permissive and has exponential complexity. Al-Ahmari *et al.* [51] propose a two-step robust deadlock control strategy for systems with unreliable and shared resources. The proposed strategy designs a common recovery subnet based on colored Petri nets for all resource failures in the Petri net model to make the system reliable. We observe that the proposed method does not ensure that the system is able to continue performing its duties, while the failed resources are under a repair and recovery process.

Kaid *et al.* [55] propose a novel three-step deadlock control strategy for fault detection and treatment of unreliable resource systems. The proposed strategy is a hybrid approach that combines neural networks with colored Petri nets for the detection and treatment of faults. We observe that the proposed method does not ensure that the system is able to continue performing its duties, while the failed resources are under a repair and recovery process. Feng *et al.* [58] develop a deadlock prevention controller for AMSs under resource failure consideration. The proposed controller ensures that the system can handle all kinds of parts continuously through any one of their paths, even if one of unreliable resources fails. Then they used a one-step look-ahead method to develop a polynomial-complexity deadlock avoidance policy (DAP). They report that during one resource failure period, the proposed controller is proved to be maximally permissive. However, we observe that the proposed controller requires the simulation-based tool to ensure the correctness, accuracy, or validity of the results.

In view of the above mentioned limitations in [16], [51], [55]–[58], in this work, a failure-safe model is proposed and, at the same time, a method for fault recovery and repair of a faulty element is presented without a single element being shut down. When there are redundant elements to replace the target elements, these redundant elements are used to do the same work as the target elements do. After a faulty target element is repaired and recovered, its failure model is automatically replaced by its new model to indicate that the corresponding elements have returned to the system.

Our contributions can be summarized as follows:

1. The proposed method does not require to introduce inhibitor arcs or enumerate reachability graphs, which leads to low computational overheads;
2. The proposed method ensures that all part types can be processed continuously no matter whether one or multiple unreliable resources fails or not;
3. Simulation-based tool is developed to ensure the correctness, accuracy, and validity of the proposed method, and comparisons are made to verify the performance of the proposed method with methods in the literature;
4. The proposed method can consider all unreliable resources in AMSs;
5. The proposed method has a simpler structure;
6. The proposed method can be applied to an unreliable complex Petri net model for AMSs.

The organization of this article is as follows. Section 2 discusses the preliminaries and fundamentals for Petri nets. In Section 3, we introduce various system states at which a fault occurs and present how to switch their models to a new model for fault-recovery and repair. In Section 4, the proposed method is explained by using an example. Section 5 provides the conclusion of the study.

II. PRELIMINARIES

We introduce some basics of Petri nets and their properties in this section, which is important in this article. We assume that readers are familiar with the fundamentals of Petri nets.

Definition 1: A basic Petri net (structure) is a four-tuple $N = (P, T, F, W)$ with finite, non-empty, and disjoint sets of P and T, where $P = P_A \cup P_R \cup P_0$ is a set of places, P_A is called the set of activity or operation places, P_R is called the set of resource places and P_0 is a set of the idle places. T is a set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$. $F \subseteq (P{\times}T){\cup}(T{\times}P)$ is called the net flow relationship defined by arcs with arrows from places to transitions or from transitions to places. W:$(P{\times}T){\cup}(T{\times}P){\rightarrow}\ \mathbb{N} = \{0, 1, 2, \ldots\}$ is a mapping assigning weight to an arc: $W(x, y) > 0$ if $(x, y) \in F$, and $W(x, y) = 0$ otherwise, where $x, y \in P \cup T$. F^{+}:$(T \times P) \rightarrow \mathbb{N}$ is the output function, and F^{-}:$(P{\times}T) \rightarrow \mathbb{N}$ is an input function. The input and output functions can be tabulated and represented by matrices indexed by place set P and transition set T. The incidence matrix F is a matrix calculated by $F = F^{+}$-F^{-}. $T = T_c{\cup}\,T_{uc}$, where T_{uc} is the set of uncontrollable transitions and T_c is the set of controllable transitions, respectively [12].

Definition 2: A mapping of $P \rightarrow \mathbb{N}$ of a Petri net N is a marking M. $M(p)$ refers to the number of tokens in place p at a marking M. A place p is marked at M if $M(P) > 0$. A subset $S \subseteq P$ is marked at M if at least one place in S is marked at M. $M(S)$ denotes the number of tokens of all places in S, i.e., $M\ (S) = \sum_{p \in S} M\ (p)$. At a marking M if $M(S) = 0$, S is said to be empty. (N, M_0) is called a net system, where M_0 is referred to as an initial marking of N.

Using a multiset (bag) or formal sum notation for space economy we ordinarily characterize markings and vectors. Accordingly, $\sum_{p \in P} M\ (p)\, p$ is used to denote vector M. For example, a marking that places four tokens in place p_4 and two tokens in place p_6 only in a net with $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ is denoted as $M = 4p_4 + 2p_6$ instead of (0,0,0,4,0,2) [12].

In general, (N, M_0), when there is no ambiguity, is explicitly called a net system or simply a net. $N = (P, T, F, W)$ is referred to as an ordinary net, denoted as $N = (P, T, F)$, if for all $f \in F$, $W(f\ ) = 1$.

The support(v) function is a set of places marked at the marking v. In the above example, support(M) $= \{p_4, p_6\}$. In an autonomous process, certain desirable properties that restrict the performance of a system are called specifications. Supervisory control aims at enforcing these pre-defined specifications. This can result in that some states are approved, while some others are prohibited. M_R is used to denote a set of markings that can be reached after firing a finite set of transitions. Let M_A be a subset of M_R for the permitted states and M_F another subset of M_R as the prohibited states. There are two groups in the set of prohibited states [8].

Definition 3 [18] (Prest and Postset): $N = (P, T, F)$ is a net where P is a finite place set, T is a finite transition set, and F is a set of arcs from a place to a transition or from a transition to a place. For all $x \in P \cup T$, we have that

a) ${}^{\cdot}x = \{y|y \in P \cup T \wedge (y, x) \in F\}$ denotes the preset of x; and
b) $x^{\cdot} = \{y|y \in P \cup T \wedge (x,y) \in F\}$ denotes the postset of x.

Similarly, given a node set $S \subset P \cup T$, S's preset and postset are defined as ${}^{\cdot}S = \cup_{x \in S}{}^{\cdot}x$ and $S^{\cdot} = \cup_{x \in S} x^{\cdot}$, respectively.

Theorem 1 [15]: Let $N = (P, T, F)$ be an acyclic Petri net (a Petri net that does not have direct circuits). If the net has no self-loop, it is pure or self-loop free, i.e., for all $t \in T$, and all $p \in P$, $|{}^{\cdot}t| \cdot |t^{\cdot}| = 0$ and $|{}^{\cdot}p|{\bullet}|p^{\cdot}| = 0$, where $|\bullet|$ denotes the cardinality of a set. Note that their incidence matrix can univocally determine the structure of a pure net. In addition, one can trivially check that an acyclic net is pure.

We consider the model of an automated manufacturing system with unreliable resources proposed in [16], where P_R denotes the set of unreliable resources in a system to be considered. A broken resource in a system can be fixed or repaired. Next, we introduce the notion of recovery subnet.

Definition 4 [17]: A PN is said to be an ordinary PN if for any arc in the net its weight is 1.

Definition 5 [17]: A transition $t \in T$ in PN is enabled in marking M if for all $p \in {}^{\cdot}t$, $M(p) \geq W(p, t)$. If a transition is enabled, it can fire. Firing an enabled transition t at marking M yields marking M' such that $M'(p) = M\ (p) - W\ (p, t) + W(t, p)$.

III. FAULT-RECOVERY AND REPAIR

A faulty state of a system is the system failure resulting from a sequence of valid state transitions. A fault is an anomalous physical condition and a manifestation of a defect in a system, which may lead to a malfunction in the performance of the system.

Definition 6: A system or plant with target elements is fault-safe if, in the case that a fault occurs to a target element, the system can continue its operations by systematically and automatically replacing the model of the target element with the model of its corresponding redundant element.

Definition 7: Failure Recovery is a process that evolves from a faulty state to a normal operational state by restoring the faulty element.

A failure in a system means that there is a fault the happens to a target element such that the system fails to perform some or all the operations. To make a system fault-tolerant, we need to deal with such a failure such that all the operations can be well performed when a failure occurs. If a system enters a failed state, a controller should immediately and automatically respond to it and lead the system to a new state such that the system can continue its operations.

We assume that there is a redundant element as an alternative resource for performing the tasks originally designed for the target element when a fault occurs.

In this study, we investigate different situations of the target non-reliable elements and provide a Petri net model that ensures the fault-safe properties by replacing a path that leads a system to a non-fault-safe state with another path that makes the system fault-safe until the former becomes safe again. In the developed Petri net, places are defined to model different status of each device and element in the system. Therefore, we need to define places to describe all possible states of the target elements and prevent the activation of the fault states represented by these places when a fault occurs. We call these places target places and redundant places for the target elements and their corresponding redundant elements, and they are denoted by p_{Ai} and p_{Ei}, respectively.

We are going to deal with failure representation problem when a failure occurs in the framework of transition/place nets. We assume that failures occur through the transition firing processes that cannot be observed, while repair and maintenance operations are done through the transition firing processes that represent observable legal behaviors.

Definition 8: Let t_{si} be a transition representing the occurrence of an unobservable and uncontrollable failure event that may occur to a target element, and t_{Ri} a transition representing the event of the repair and maintenance operation for a failed target element, and such an event is observable and controllable.

Let e_F denote a failure and e_R the repair and recovery event for a failed element. When $e_F = 1$ and $e_R = 0$, transition t_{si} is enabled; while $e_F = 0$ and $e_R = 1$, transition t_{Ri} is enabled.

Let $\Delta = \{f_1, f_2, \ldots\}$ be the set of faults that might occur in the system. We model a fault $fi \in \Delta$ by an unobservable fault transition $\tau_{si} \in T_f$ with $T_f = \{\tau_1, \tau_2, \ldots, \tau_F\} \subseteq T_{uo}$. Assume

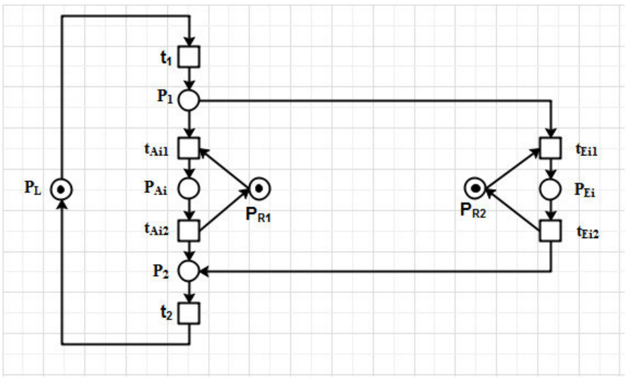


FIGURE 1. The Petri net model to a target element and its corresponding redundant element.

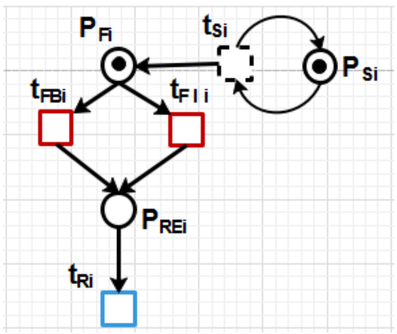


FIGURE 2. The Petri net model of a recovery subnet.

that the set of transitions T in a Petri net is divided into two disjoint subsets, i.e., $T = T_o \cup T_{uo}$, where T_{uo} represents the set of unobservable transitions and T_o represents the set of observable transitions. Let $t_{si} \in T_{uo} = \{t_{si}|\text{i} = 1, 2, \ldots, n\}$, and $t_{Ri} \in T_o = \{t_{Ri}|\text{i} = 1, 2, \ldots, n\}$, where n is the number of target elements that are failure-prone, and there is a redundant element for each target element.

Consider the Petri net model in Fig. 1, which represents a system with two similar elements: target element represented by the place p_{Ai} and its corresponding redundant element represented by p_{Ei}, respectively. In this model, the redundant element is connected to the system, and therefore it is a part of the system. However, the aim of the model is to make only one of these two elements operate at a time. Since operating only one of them is enough for the system to run properly, the purpose of using the redundant is to replace the target element only when there is a fault to the target element. Thus, it needs a controller that can automatically activate the redundant element in the event of a failure and, at the same time, deactivate it when the target element has been repaired and recovered.

When there is a fault to the target element, the controller responsible for activating and deactivating transitions to make the system act properly is presented in Fig. 2. In the subnet p_{Fi} represents that a failure occurs to a target element and it is marked when the target element fails. If p_{REi} is marked, it denotes that the target element is under maintenance.

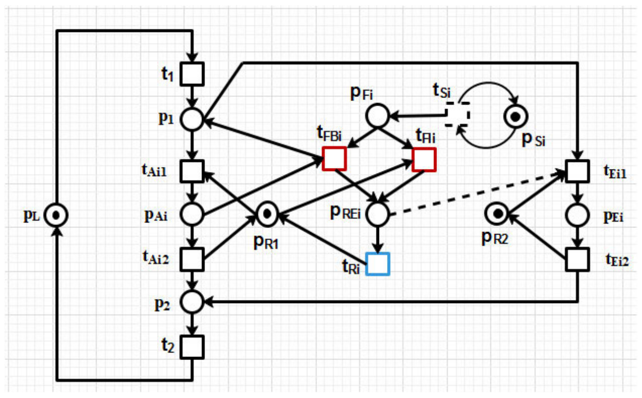


FIGURE 3. The Petri net model with added recovery subnet.

t_{FIi} and t_{FBi} are transitions that indicate the occurrence of failure to the target element, t_{Si} is the transition that describes the time to failures, and t_{Ri} is the transition that represents repair and recovery of a target element.

The recovery subnet can be added to the Petri net model as shown in Fig. 3.

Definition 9: An arc connecting a place p to a transition t, denoted as $p \xrightarrow{w(f)} t$, is said to be a test arc if a token can be released from p through t into $t^{\cdot}$ when $M(p) \geq w(f)$ and the marking $M(p)$ remains unchanged.

Definition 10: Let $p_{Ai} \in P_R$ be an unreliable element or resource in a system and let $H(p_{Ai})$ be the set of places that hold p_{Ai} called the holders of p_{Ai} indicated by $H(p_{Ai}) = \{p|p \in P_A, p \in {}^{\cdot\cdot}p_{Ai} \cap P_A \neq \emptyset\}$.

Definition 11 [16], [51]: Let $p_{Ai} \in P_R$ be an unreliable element. A recovery subnet of p_{Ai} is a Petri net $N_{FRi} = (\{p_i, p_{REi}\}, \{t_{FRi}, t_{Ri}\}, F_{FRi})$, where $p_i \in \{p_{Ai}\} \cup H(p_{Ai})$, p_{REi} is called the recovery place of all p_i. Transitions t_{FRi} and t_{Ri} indicate that an unreliable resource p_{Ai} fails in p_i and recovers through p_{RE} and $F_{FRi} = \{(p_i, t_{FRi}), (t_{FRi}, p_{REi}), (p_{REi}, t_{Ri}), (t_{Ri}, p_i)\}$, and an unreliable element p_{Ai} may fail when it is idle or in a busy state. Thus, we define $P_X = \{p_{Ai}\} \cup H(p_{Ai})$ as a set of places. Adding one recovery subnet for all $p_i \in P_X$, results in an unreliable Petri net defined as $(N_C, M_{Co}) = (N, M_o) \parallel (N_{FRi}, M_{FRio})$ that is the composition of (N, M_o) and (N_{FRi}, M_{FRio}), where (N_{FRi}, M_{FRio}) is the recovery subnet, where $M_{FRio}(p_i) \geq 0$ and $M_{rio}(p_{REi}) = 0$.

For a system with failure-prone target elements, there is a failure repairing, maintenance and recovery process. To make such a system fault-safe, controllers should be designed for different cases in order to handle any occurrence of failures and the repairing, maintenance and recovery process of a failure as well.

A. THE CASE WITH NO TARGET ELEMENT FAILED

When there is no failure, a failure-prone target element is in operation, while its corresponding redundant element is in its idle state. In this case, we model the system by using Petri net as shown in Fig. 4, where places p_{R1} and p_{R2} are the resource places for a target element and its corresponding redundant element, respectively, with $M(p_{R1}) = 1$ and $M(p_{R2}) = 1$,

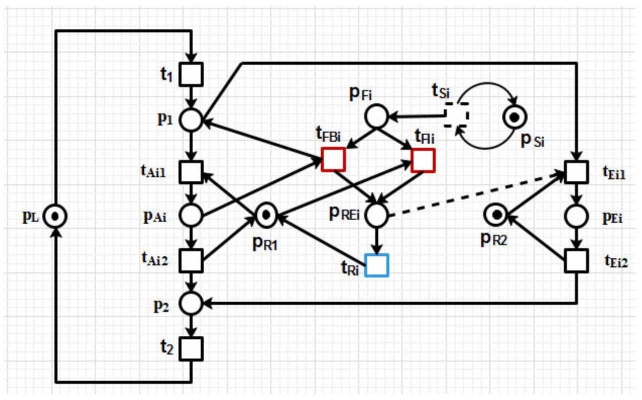


FIGURE 4. The Petri net model with no target element failed.

TABLE 1. Description of places and transitions in fig. 4.

Place	Description	Transition	Event
p_L	Idle place	t_1	Start process
p_1	The element (target or redundant) is ready to perform the operation	t_2	End process
p_2	Completion of the process by the element (target or redundant)	t_{Ai1}	Start an operation performed by the target element
p_{Ai}	The operation performed by the target element	t_{Ai2}	Complete an operation performed by the target element
p_{Ei}	The operation performed by the redundant element	t_{Ei1}	Start an operation performed by the redundant element
p_{R1}	Resource place for the target element	t_{Ei2}	Complete an operation performed by the redundant element
p_{R2}	Resource place for the redundant element	t_{Si}	Time to failures
p_{Fi}	A failure occurs to the target element	t_{Ri}	Repair of target element
p_{REi}	The target element is under maintenance	t_{FIi}	A failure occurs when the target element is idle
p_{si}	Place to failures	t_{FBi}	A failure occurs when the target element is in operation

denoting that their capacity is one. Note that the place that models the recovery and repair of the target element is not marked, i.e., $M(p_{REi}) = 0$, which makes the redundant element unable to be put into operation. The meaning for the places and transitions in Fig. 4 is presented in Table 1.

Theorem 2: The Petri net (N_{FR}, M_{FRo}) with unreliable element is live if an unreliable element does not fail.

Proof: A failure of the unreliable element can be observed by the failure of firing transition $t_{Ai} \in T_o$. Since there is no failure to the unreliable element, the net is live and the transition $t_{Ai} \in T_o$ is also live. This means that there are no dead transitions.

B. THE CASE WITH ONE TARGET ELEMENT FAILED

A failure can happen in one of the two possible conditions of a target element: a failure can occur when a target element is

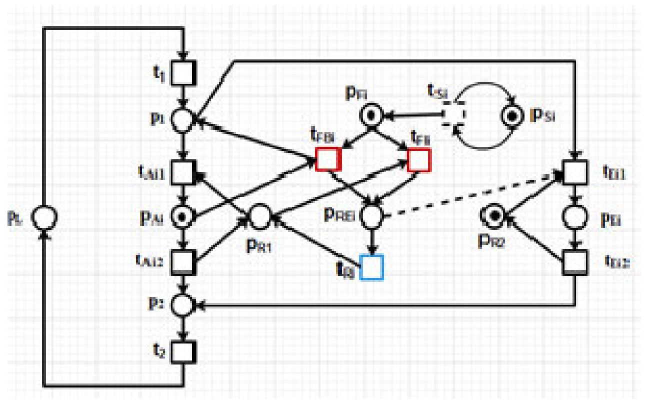


FIGURE 5. The Petri net model for the case that a failure occurs to an element in operation.

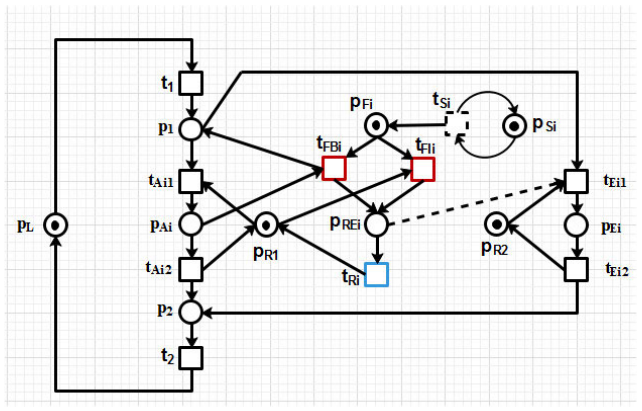


FIGURE 6. The Petri net model for a failure happened to idle target element.

in operation or is idle. These two situations are discussed as follows, respectively.

1) FAILURE OF AN ELEMENT IN OPERATION

If the target machine breaks while processing a task, this element cannot continue its operation. Therefore, the task must be switched to the redundant element after it is activated to perform the task, which is done as follows: when a failure occurs to a target element that is in operation, this element cannot continue its operation. Thus, the Petri net should describe this fact by disabling some transitions and changing the marking in some places. At the same time, to make the system failure-safe, it needs to activate its corresponding redundant element by automatically enabling some transitions and changing the marking of some places in the Petri net model. Therefore, we need to synthesize a supervisor to the Petri net model such that the system can act in a correct way. For this case, the Petri net model is shown in Fig. 5. As shown in Fig. 5, when the target element fails, when it is in operation. Thus, p_{R1} representing the target element is not marked, while the operation place p_{Ai} is marked, i.e., $M(p_{R1}) = 0$ and $M(p_{Ai}) = 1$. To make the system failure-safe, we need to automatically remove the token in p_{Ai} and activate the corresponding redundant element such that the operation can be continued.

The Petri net model shown in Fig. 5 presents a policy to control the system such that the system can continue its operations correctly. At the marking shown in Fig. 5, transition t_{Si} is enabled and its firing deposits a token into p_{Fi}, which in turn enables t_{FBi}. As a result, we have that, for all $p \in {}^{\cdot}t_{FBi}$, $M(p) > 0$ such that t_{FBi} is enabled. Firing t_{FBi} indicates the occurrence of a failure to the target element. This takes the token in p_{Ai} away and deposits a token into both p_1 and p_{REi}. Then, the controller takes the token in p_{Ai} away to stop its operation, and adds a token to place p_{Ei} to activate the corresponding redundant element for performing the task that would be performed by the target element by firing t_{FBi}. With p_{REi} being marked, the target element is under repair, i.e., the repair mode is activated for the target element.

After the target element is repaired and returns to work, the transition t_{Ri} is enabled and its firing deposits a token into p_{R1}. This process is as follows. After some time, the token in p_{REi} triggers t_{Ri} since, at this time, for all $p \in {}^{\cdot}t_{Ri}$, $M(p) > 0$ holds. The firing of t_{Ri} releases a token into p_{R1} that represents the target element, such that the target element can perform the tasks due to that, for all $p \in {}^{\cdot}t_{Ai1}$, $M(p) > 0$, since $M(p_{REi}) > 0$.

Suppose that the system consists of a self-loop between t_{Si} and p_{Si}. Then transition t_{Si} releases a token to place p_{Fi} when a failure occurs to the target element according to a given rate based on statistics analysis. The above proposed control policy can ensure the safe operations of the system every time a target element fails.

2) FAILURE TO AN IDLE TARGET ELEMENT

Suppose that a target element has just finished a task and suddenly it fails before it starts performing the next task. In this case, we can think that the target element fails when it is idle. For this case, we also need to control the system such that the tasks to be processed by this target element can be processed without interruption.

For this case, since the target element is idle, place p_{R1} is marked as shown in Fig. 6, i.e, $M(p_{R1}) > 0$. In this case, transition t_{Si} is triggered to indicate that the target element (p_{R1}) has failed. Its firing deposits a token into p_{Fi}. As for all $p \in {}^{\cdot}t_{FIi}$, $M(p) > 0$, t_{FIi} fires and releases a token into p_{REi}. This activates the corresponding redundant element by enabling the transition t_{Ei1}.

In this case, when the target element is recovered, the same procedure as in the case that a failure occurs to an element that is in operation can be followed in order to activate the target element and deactivate the redundant element. When the system has finished some tasks by the redundant element and more tasks need to be processed while the target element is still in a repairing state, there is still a token in p_{REi}, which activates the redundant element. Note that the dashed arc between a place p and a transition t is called a test arc. In Definition 11, p_{REi} is said to be the recovery place of $p_i \in P_X$. Transitions t_{FRi} and t_{Ri} indicate that an unreliable

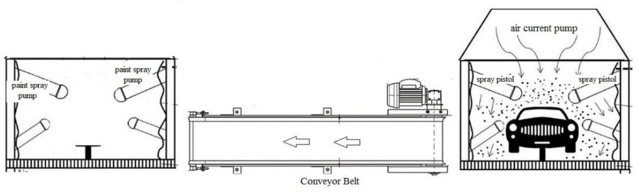


FIGURE 7. A repainting system.

target resource p_{Ai} fails in p_i and recovers through p_{REi}, respectively.

Assumption 1: Transitions t_{Si}, t_{FBi}, t_{FIi}, and t_{Ri} of the subnet model are reliable (i.e., transitions that do not fail to fire whenever enabled).

Theorem 3: The Petri net (N_{FR}, M_{FRo}) with an unreliable element can be recovered by the recovery subnet and become live if the target element p_{Ai} fails, regardless of whether it is idle or in operation.

Proof: When the target element fails, the processing of a task modeled by p_{Ai} cannot be continued such that the token in p_{Ai} should be removed away. At the same time, there is no token in p_{R1} representing the availability of the target element. Thus, the transitions t_{Ai1} and t_{Ai2} are disabled. When the target element fails, p_{REi} gets a token, representing the target element is under repairing. Then, transitions $t_{Ei} \in T_o$ is enabled. Enabling $t_{Ei} \in T_o$ means that the redundant element is activated to replace the failed unreliable target element. Thus, $t_{Ei} \in T_o$ is used to replace $t_{Ai} \in T_o$ in the model. No matter whether the unreliable target element is idle or busy, the recovery subnet (N_{FR}, M_{FRo}) will always activate the redundant element when the unreliable element fails. Thus, Petri net (N_{FR}, M_{FRo}) always remains live under the controlled fault recovery and repair process.

IV. A PRACTICAL EXAMPLE

It is a polishing system for the repainting of some metal objects and it is shown in Fig. 7. There are two air current pumps (AP) with grains of sand for removing rust and the previous coating from the surface of the metal, one of them used as redundancy; a washing machine (WM) with more than one spray pistol to remove oil and grease from the metal surface; a conveyor belt (CB) to transport the metal from the polishing and washing area to the place of painting; and two paint spray pumps (SP) to spray paint onto the surface of the metal, one of them used as redundancy. The Petri net model for this system is shown in Fig. 8. Table 2 presents the meaning of places and transitions in Fig. 8.

Assume that the main air pump, the main motor for driving the conveyor belt, and the main paint spray pump are failure-prone. When one of them fails, the following actions should be taken.

- When the main air pump fails, it should be replaced by the spare air pump until it is repaired and returned to work.

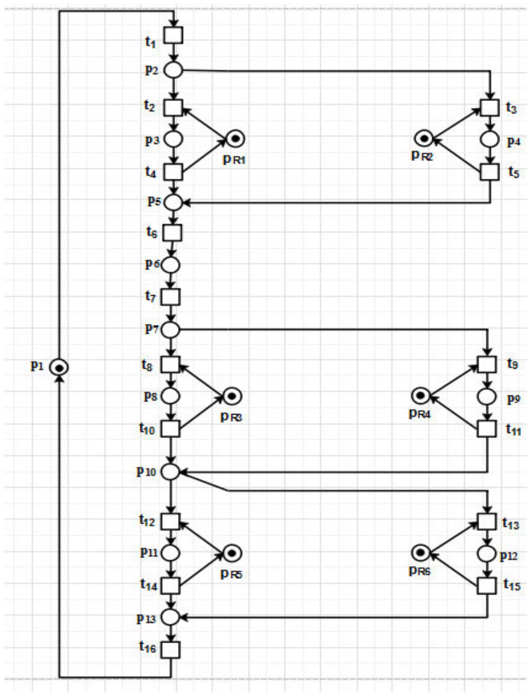


FIGURE 8. The PN model for the example.

TABLE 2. Description of places and transitions of the example.

Place	Description	Transition	Event
p_1	System off	t_1	Insert the metal into the polishing and washing area.
p_2	Metal in the polishing and washing area	t_2 & t_3	Start pumping air currents on the metal surface to remove rust and residues from the previous paint.
p_3 &p_4	Metal under air currents	t_4 & t_5	Air stream pump stops
p_5	Waiting for the removal of impurities, oils, and grease	t_6	Start the washing process
p_6	The surface of the metal under the laundry	t_7	Completion of the washing process
p_7	Waiting for the metal to be transported to the painting area	t_8 & t_9	Start the transfer of the metal by the conveyor belt to the painting area
p_8 &p_9	Metal on a conveyor belt	t_{10} & t_{11}	Stop the conveyor belt and the metal arrives the painting area
p_{10}	Metal waiting for the painting process	t_{12} & t_{13}	Start the spraying process
p_{11}& p_{12}	The surface of the metal under the coating process	t_{14} & t_{15}	Stop the painting process
p_{13}	Metal waiting to be taken out from the painting area	t_{16}	Take the metal out of the painting area and return the system to the starting state.

- When a failure occurs to the main motor of the conveyor belt, the system should switch to the redundant motor until the main motor is repaired.
- The replacement of the spare paint spray pump should be done, when there is a fault to the main paint spray pump.

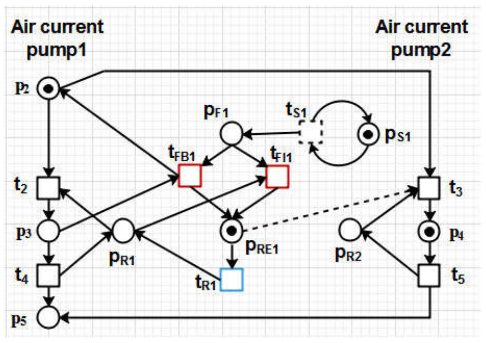


FIGURE 9. The Petri net for realizing the control when the air pump in operation fails.

TABLE 3. The expected cases for different support(*M*).

Marking M with support(M)	Description using the model shown in Fig.10
support(M)= (p_{R3}, p_{R5})	A failure occurs to the main air pump only, and it is replaced by its redundant air pump.
support(M)= (p_{R5})	Two failures occur to the main air pump and the main motor of the conveyor belt, and they are replaced by the corresponding redundant elements.
support(M)= (p_{R3})	Two failures occur to the main air pump and the main spray pump, and they are replaced by the corresponding redundant elements.
support(M)= (p_{R1}, p_{R3})	A failure occurs to the main spray pump, and it is replaced by the redundant spray pump.

If the main air pump fails, its operation has to be stopped and the redundant one is put into operation until the main air pump is repaired. Therefore, in this case, the controller should prevent place p_3 from getting a token if there is no token in it, representing that the mental is ready for being processed or it should remove the token from it if there is a token in it.

If the main pump fails before it starts to work (p_3 is not marked and p_{R1} is marked), then t_{s1} is triggered and its firing releases a token to p_{F1}, thus enabling t_{FI1}, which in turn removes the token from p_{R1} and delivers the token into p_{RE1}. In this way, the redundant air pump is activated to replace the main air pump and, at the same time, the main air pump is under repair and maintenance.

If the main pump fails when it is in operation (p_3 is marked and, p_{R1} is not marked), t_{s1} fires and releases a token to p_{F1}, enabling t_{FB1}, which in turn removes the token in p_3 and puts a token into both p_2 and p_{RE1}. In this way, the redundant air pump is activated and the main air pump is put under repair and maintenance. The Petri net for realizing such a process is given in Fig. 9.

When the main air pump is repaired and recovered, t_{R1} fires and puts a token into p_{R1}, which activates the main air pump and, at the same time, turns off the redundant air pump.

Similarly, when the main-belt conveyor motor or the main paint spray pump at different states fails, we can present similar Petri net models to control the system such that the system is failure-safe just as the cases for the air pump.

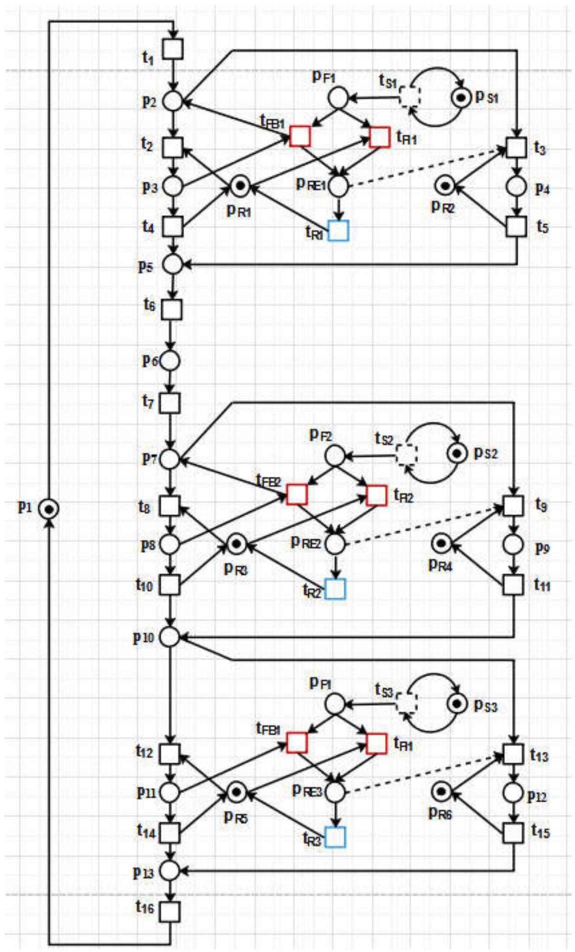


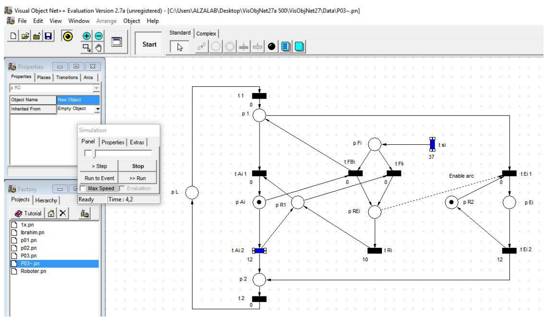
FIGURE 10. The Petri net model for the final controller the application example.

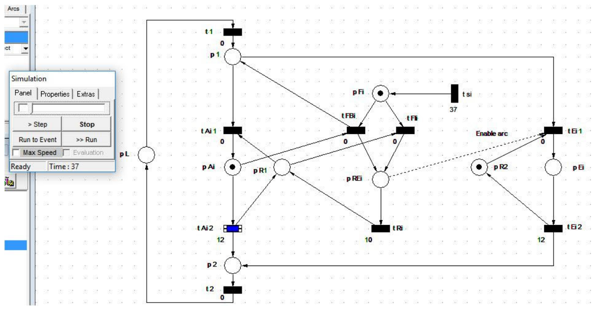
As above discussed, we can achieve the fault-recovery and repair controller by synchronizing the fault-recovery and repair models and the major plant model. The final controlled Petri net model of the application example is shown in Fig. 10.

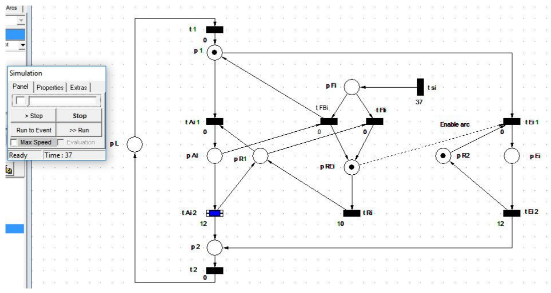
The entire controlled system is shown in Fig. 10. For this controlled system, it is fault-safe with respect to a failure of any unreliable element (the main air pump, the main motor of the conveyor belt, or the main industrial spray pump). For the controlled system shown in Fig. 10, we consider marking M with support(M) = (p_{R1}, p_{R3}, p_{R5}) and, for other marking M with different Support(M), the controller is explained in Table 3.

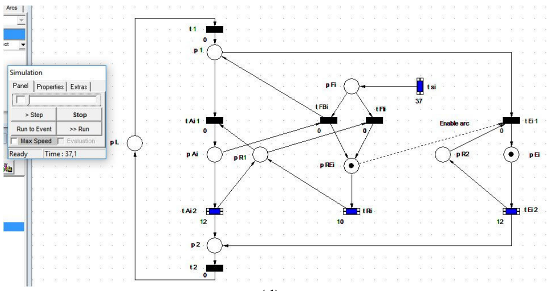
V. SIMULATION OF THE APPLICATION EXAMPLE

Simulation is an important way used to evaluate performance and verify the validity of a proposed method. Visual Object Net ++ is a software tool for simulating, evaluating and modelling discrete event systems based on Petri net models. With the above exemple, to validate the proposed method, simulation is carried out by using an application example based on Visual Object Net ++.


(a)


(b)


(c)


(d)

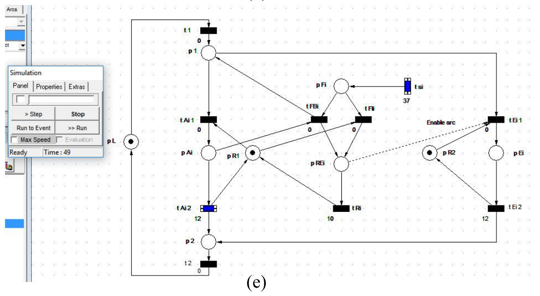

(e)

FIGURE 11. **(a) The target element before failure. (b) A failure occurs to the target element. (c) The target element after failure. (d) The target element under repair. (e) The target element after being repaired.**

In the simulation, we consider failures happened to the target elements after a period of operation for the example

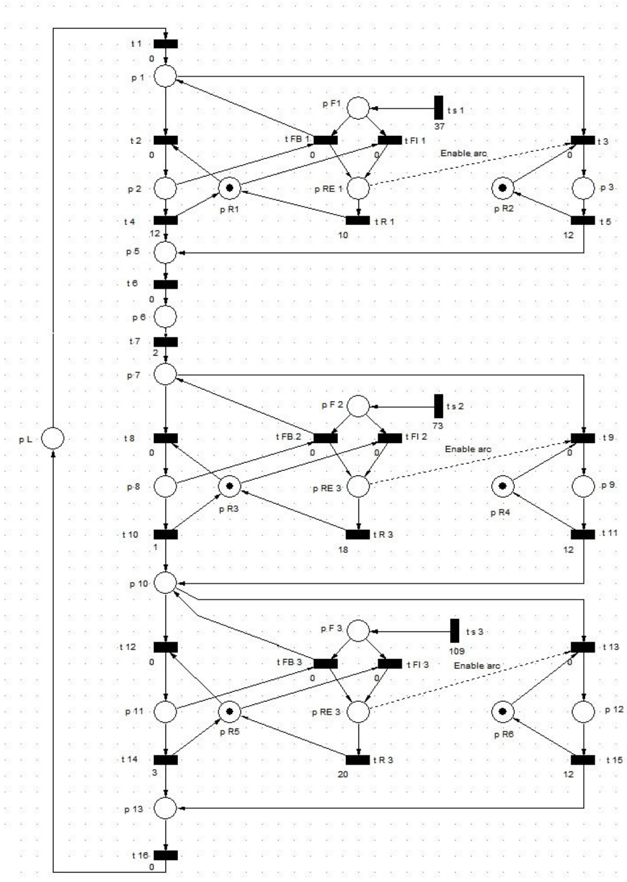


FIGURE 12. **The Petri net model for simulation using the application example.**

system over different time periods. System task performance is transferred to the redundant element. When a fault occurs to a target element, the failed target element is replaced by its redundant element and the redundant element continues to perform tasks before the target element is repaired. After the failed target element is recovered, the redundant element is turned off and the processing of tasks is returned to the target element. Fig. 11 shows different states for a target element before and after the failure, and during repair and after repair.

Assume that a failure occurs to some elements in the system at different time during the estimated time period. The simulation for the application example is based on the model given in Fig. 12.

In the model given in Fig. 12, the transition t_{si} represents the self-loop for triggering failures in the simulation. The time periods for the simulation are given in Table 4.

To verify, validate, and make comparisons of the time performance criteria for the proposed method, we have compared the simulation models with the studies by Liu *et al.* [16], Al-Ahmari *et al.* [51], and Kaid *et al.* [55]. After running and simulating the models, the results are summarized in Table 5. Table 5 and Fig. 13 show the results in terms of utilization of the resources, throughput (finished cars), and total time in system (throughput time for each car). From the viewpoint of resource utilization, the proposed

TABLE 4. The resource time for the simulation.

Transition (t)	Time duration (min)	Description
t_4 & t_5	60	The operating time of the main air pump and the redundant air pump.
t_7	10	Operating time of the washing machine to remove oil and grease from the metal surface
t_{10} & t_{11}	5	Operating time of belt conveyor using one of the two motors (main motor, redundant motor)
t_{14} & t_{15}	15	Operating time for both the main spray pump and the redundant spray pump.

TABLE 5. Time performance comparison with the existing methods.

Performance	Liu et al. [16]	Al-Ahmari et al. [51]	Kaid et al. [55]	The proposed method
AP utilization%	63.41	64.64	64.64	65.45
CB utilization%	5.28	5.39	5.39	5.45
SP utilization%	15.34	16.16	16.16	16.36
Throughput (Cars)	15	15	15	16
Throughput time (min/ car)	94.62	92.83	92.83	91.67

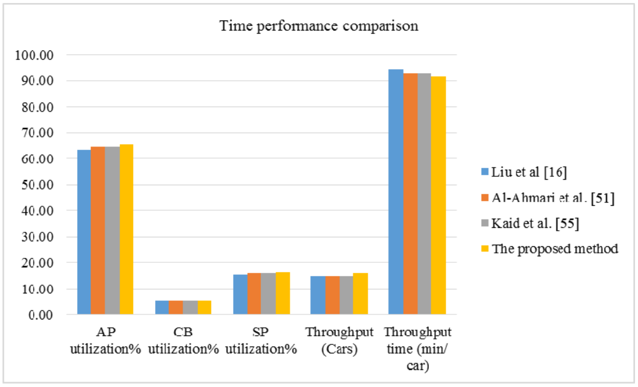


FIGURE 13. Comparison of the proposed method with the existing methods.

model achieves better performance than the other approaches in the literature, as shown in Fig. 13. With respect to throughput, the proposed method can achieve a higher throughput than other approaches as shown in Fig. 13. Finally, in terms of total time in system, the proposed method can achieve less total time in system than other approaches as shown in Fig. 13. Thus, the proposed method is accurate and can potentially be implemented to other instances.

VI. CONCLUSION

This work proposes an approach for investigating the behavior of faults to the resources in AMS that are prone to failure. We construct a redundant Petri net model for a target element that is prone to failure in order to synthesize a fail-safe controller for the system and make it able to continue performing its duties when a target element fails and is under repair or maintenance. The proposed method is tested using a practical example. The results are compared with the studies in the literature. The main advantages of the proposed method are as follows: (1) The proposed method achieves a good performance and performs better than the studies in the literature [16], [51], and [55]. (2) The proposed method ensures that all part types can be processed continuously no matter whether one or multiple unreliable resources fails or not. (3) The proposed method can handle all unreliable resources in AMSs. (4) The proposed method can be applied to an unreliable complex Petri net model for AMSs. (5) Simulation-based tool is developed to ensure the correctness, accuracy, and validity of the proposed method.

Our future work is to expand this method to the systems with uncontrollable events.

REFERENCES

[1] R. J. Patton, F. J. Uppal, and C. J. Lopez-toribio, "Soft computing approaches to fault diagnosis for dynamic systems: A survey," *IFAC Proc. Volumes*, vol. 33, no. 11, pp. 303–315, Jun. 2000.
[2] 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, Jan. 1996.
[3] Q. Wen, R. Kumar, J. Huang, and H. Liu, "A framework for fault-tolerant control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 8, pp. 1839–1849, Sep. 2008.
[4] H. Darabi, M. A. Jafari, and A. L. Buczak, "A control switching theory for supervisory control of discrete event systems," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 131–137, Feb. 2003.
[5] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, vol. 2. Berlin, Germany: Springer, 2006.
[6] B. H. Krogh and L. E. Holloway, "Synthesis of feedback control logic for discrete manufacturing systems," *Automatica*, vol. 27, no. 4, pp. 641–651, Jul. 1991.
[7] A. Giua and F. DiCesare, "Supervisory design using Petri nets," in *Proc. 30th IEEE Conf. Decis. Control*, Dec. 1991, pp. 92–97.
[8] A. Dideban and H. Alla, "Reduction of constraints for controller synthesis based on safe Petri nets," *Automatica*, vol. 44, no. 7, pp. 1697–1706, Jul. 2008.
[9] P. Nazemzadeh, A. Dideban, and M. Zareiee, "Fault modeling in discrete event systems using Petri nets," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–19, Jan. 2013.
[10] P. Nazemzadeh, A. Dideban, and M. Zareiee, "Fail-safe modeling of discrete event systems using Petri nets," *Int. J. Electr. Comput. Eng.*, vol. 5, no. 1, pp. 24–29, 2011.
[11] M. P. Cabasino, A. Giua, M. Pocci, and C. Seatzu, "Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems," *Control Eng. Pract.*, vol. 19, no. 9, pp. 989–1001, Sep. 2011.
[12] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer, 2009.
[13] M. V. Ioradache and P. J. Antsaklis, "Resilience to failures and reconfigurations in the supervision based on place invariants," in *Proc. Amer. Control Conf.*, vol. 5, 2004, pp. 4477–4482.
[14] P. E. Miyagi and L. A. M. Riascos, "Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets," *Control Eng. Pract.*, vol. 14, no. 4, pp. 397–408, Apr. 2006.
[15] A. Giua, C. Seatzu, and D. Corona, "Marking estimation of Petri nets with silent transitions," *IEEE Trans. Autom. Control*, vol. 52, no. 9, pp. 1695–1699, Sep. 2007.
[16] 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.
[17] M. C. Zhou and N. Q. Wu, *System Modeling and Control With Resource-Oriented Petri Nets*. Boca Raton, FL, USA: CRC Press, 2009.
[18] Y. Xu, Y. Du, L. Qi, W. Luan, and L. Wang, "A logic Petri net-based model repair approach by constructing choice bridges," *IEEE Access*, vol. 7, pp. 18531–18545, Jan. 2019.
[19] G. Zhu, Z. Li, and N. Wu, "Model-based fault identification of discrete event systems using partially observed Petri nets," *Automatica*, vol. 96, pp. 201–212, Oct. 2018.
[20] X. Cong, M. P. Fanti, A. M. Mangini, and Z. Li, "On-line verification of current-state opacity by Petri nets and integer linear programming," *Automatica*, vol. 94, pp. 205–213, Aug. 2018.

[21] X. Li, Z. Yu, Z. Li, and N. Wu, "Group consensus via pinning control for a class of heterogeneous multi-agent systems with input constraints," *Inf. Sci.*, vol. 542, pp. 247–262, Jan. 2021, doi: 10.1016/j.ins.2020.05.085.

[22] L. Yang, Z. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Wu, "On multiplexity-aware influence spread in social networks," *IEEE Access*, vol. 8, pp. 106705–106713, 2020.

[23] L. Yang, Z. Li, and A. Giua, "Containment of rumor spread in complex social networks," *Inf. Sci.*, vol. 506, pp. 113–130, Jan. 2020.

[24] X. Li, E. S. Abouel Nasr, A. Mohammed El-Tamimi, and Z. Li, "Adaptive consensus of two coupled heterogeneous networked systems with bidirectional actions," *IEEE Access*, vol. 8, pp. 35832–35841, 2020.

[25] X. Zan, Z. P. Wu, C. Guo, and Z. Yu, "A Pareto-based genetic algorithm for multi-objective scheduling of automated manufacturing systems," *Adv. Mech. Eng.*, vol. 12, no. 1, pp. 1–15, 2020, doi: 10.1177/1687814019885294.

[26] L. Li, F. Basile, and Z. Li, "An approach to improve permissiveness of supervisors for GMECs in time Petri net systems," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 237–251, Jan. 2020.

[27] Y. Liu, K. Cai, and Z. Li, "On scalable supervisory control of multi-agent discrete-event systems," *Automatica*, vol. 108, Oct. 2019, Art. no. 108460, doi: 10.1016/j.automatica.2019.06.012.

[28] Q. Chen, L. Yin, N. Wu, M. A. El-Meligy, M. A. F. Sharaf, and Z. Li, "Diagnosability of vector discrete-event systems using predicates," *IEEE Access*, vol. 7, pp. 147143–147155, 2019.

[29] D. Wang, X. Wang, and Z. Li, "Nonblocking supervisory control of state-tree structures with conditional-preemption matrices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3744–3756, Jun. 2020.

[30] 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.

[31] Y. Hu, Z. Ma, and Z. Li, "Design of supervisors for active diagnosis in discrete event systems," *IEEE Trans. Autom. Control*, early access, Jan. 28, 2020, doi: 10.1109/TAC.2020.2970011.

[32] D. A. Zaitsev, "Verification of computing grids with special edge conditions by infinite Petri nets," *Autom. Control Comput. Sci.*, vol. 47, no. 7, pp. 403–412, Dec. 2013, doi: 10.3103/S0146411613070262.

[33] D. A. Zaitsev, V. G. Sarbei, and A. I. Sleptsov, "Synthesis of continuous-valued logic functions defined in tabular form," *Cybern. Syst. Anal.*, vol. 34, no. 2, pp. 190–195, Mar. 1998, doi: 10.1007/BF02742068.

[34] C. Gu, Z. Li, N. Wu, M. Khalgui, T. Qu, and A. Al-Ahmari, "Improved multi-step look-ahead control policies for automated manufacturing systems," *IEEE Access*, vol. 6, pp. 68824–68838, 2018.

[35] C. Gu, X. Wang, and Z. Li, "Synthesis of supervisory control with partial observation on normal state-tree structures," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 984–997, Apr. 2019.

[36] C. Gu, X. Wang, Z. Li, and N. Wu, "Supervisory control of state-tree structures with partial observation," *Inf. Sci.*, vol. 465, pp. 523–544, Oct. 2018.

[37] K. Xu, W. Pedrycz, Z. Li, and W. Nie, "High-accuracy signal subspace separation algorithm based on Gaussian kernel soft partition," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 491–499, Jan. 2019.

[38] D. You, S. Wang, and M. Zhou, "Synthesis of monitor-based liveness-enforcing supervisors for S^3PR with ξ-resources," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 6, pp. 967–975, Jun. 2015.

[39] 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.

[40] K. Xing, M. Chu Zhou, H. Liu, and F. Tian, "Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 1, pp. 188–199, Jan. 2009.

[41] 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.

[42] K. Y. Xing, B. S. Hu, and H. X. Chen, "Deadlock avoidance policy for Petri-net modeling of flexible manufacturing systems with shared resources," *IEEE Trans. Autom. Control*, vol. 41, no. 2, pp. 289–295, Feb. 1996.

[43] 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, Sep. 2015, Art. no. 537893.

[44] S. Zhou, Z. Yu, E. S. A. Nasr, H. A. Mahmoud, E. M. Awwad, and N. Wu, "Homomorphic encryption of supervisory control systems using automata," *IEEE Access*, vol. 8, pp. 147185–147198, 2020, doi: 10.1109/ACCESS.2020.3014217.

[45] 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.

[46] S. Wang, W. Duo, X. Guo, X. Jiang, D. You, K. Barkaoui, and M. Zhou, "Computation of an emptiable minimal siphon in a subclass of Petri nets using mixed-integer programming," *IEEE/CAA J. Autom. Sinica*, early access, Jun. 2, 2020, doi: 10.1109/JAS.2020.1003210.

[47] N. Du, H. Hu, and M. Zhou, "Robust deadlock avoidance and control of automated manufacturing systems with assembly operations using Petri nets," *IEEE Trans. Autom. Sci. Eng.*, early access, Apr. 30, 2020, doi: 10.1109/TASE.2020.2983672.

[48] 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, May 2019.

[49] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on Petri nets with data operations," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 251–260, Jan. 2018.

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

[51] A. Al-Ahmari, H. Kaid, Z. Li, and R. Davidrajuh, "Strict minimal siphon-based colored Petri net supervisor synthesis for automated manufacturing systems with unreliable resources," *IEEE Access*, vol. 8, pp. 22411–22424, 2020.

[52] F. Yang, N. Wu, Y. Qiao, and R. Su, "Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 270–280, Jan. 2018.

[53] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajuh, "Intelligent colored token Petri nets for modeling, control, and validation of dynamic changes in reconfigurable manufacturing systems," *Processes*, vol. 8, no. 3, p. 358, Mar. 2020.

[54] N. Wu and M. Zhou, "Deadlock resolution in automated manufacturing systems with robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 3, pp. 474–480, Jul. 2007.

[55] H. Kaid, A. Al-Ahmari, E. A. Nasr, A. Al-Shayea, A. K. Kamrani, M. A. Noman, and H. A. Mahmoud, "Petri net model based on neural network for deadlock control and fault detection and treatment in automated manufacturing systems," *IEEE Access*, vol. 8, pp. 103219–103235, 2020.

[56] M. C. Zhou and F. Dicesare, "Adaptive design of Petri net controllers for error recovery in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 963–973, 1989.

[57] 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.

[58] Y. Feng, K. Xing, M. Zhou, H. Chen, and F. Tian, "Polynomial-complexity robust deadlock controllers for a class of automated manufacturing systems with unreliable resources using Petri nets," *Inf. Sci.*, vol. 533, pp. 181–199, Sep. 2020.

[59] L. Bai, N. Wu, Z. Li, and M. Zhou, "Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 10, pp. 1456–1467, Oct. 2016.

[60] S. Zhang, N. Wu, Z. Li, T. Qu, and C. Li, "Petri net-based approach to short-term scheduling of crude oil operations with less tank requirement," *Inf. Sci.*, vol. 417, pp. 247–261, Nov. 2017.

[61] N. Q. Wu, M. C. Zhou, L. P. Bai, and Z. W. Li, "Short-term scheduling of crude oil operations in refinery with high fusion point oil and two transportation pipelines," *Enterprise Inf. Syst.*, vol. 10, no. 6, 581–610, May 2016.

[62] Y. Qiao, N. Wu, F. Yang, M. Zhou, and Q. Zhu, "Wafer sojourn time fluctuation analysis of time-constrained dual-arm cluster tools with wafer revisiting and activity time variation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 4, pp. 622–636, Apr. 2018.