IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received April 28, 2021, accepted May 19, 2021, date of publication May 31, 2021, date of current version June 18, 2021.
Digital Object Identifier 10.1109/ACCESS.2021.3084995

Deadlock Control and Fault Detection and Treatment in Reconfigurable Manufacturing Systems Using Colored Resource-Oriented Petri Nets Based on Neural Network

HUSAM KAID[1,2], ABDULRAHMAN AL-AHMARI[1,2], (Member, IEEE), ZHIWU LI[3], (Fellow, IEEE), AND WADEA AMEEN[4]

[1]Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[2]Raytheon Chair for Systems Engineering (RCSE Chair), Advanced Manufacturing Institute, King Saud University, Riyadh 11421, Saudi Arabia
[3]Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, China
[4]Industrial Engineering Department, College of Engineering and Architecture, Al-Yamamah University, Riyadh 11512, Saudi Arabia

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

This work was supported by the Raytheon Chair for Systems Engineering.

ABSTRACT A reconfigurable manufacturing system (RMS) means that it can be reconfigured and become more complex during its operation. In RMSs, deadlocks may occur because of sharing of reliable or unreliable resources. Various deadlock control techniques are proposed for RMSs with reliable and unreliable resources. However, when the system is large-sized, the complexity of these techniques will increase. To overcome this problem, this paper develops a four-step deadlock control policy for the detection and treatment of faults in an RMS. In the first step, a colored resource-oriented timed Petri net (CROTPN) is designed for rapid and effective reconfiguration of the RMS without considering resource failures. In the second step, ''sufficient and necessary conditions'' for the liveness of a CROTPN are introduced to guarantee that the model is live. The third step considers the problems of failures of all resources in the CROTPN model and guarantees that the model is reliable by designing a common recovery subnet and adding it to the obtained CROTPN model at the second step. The fourth step designs a new hybrid method that combines the CROTPN with neural networks for fault detection and treatment. A simulation is performed using the GPenSIM tool to evaluate the proposed policy under the RMS configuration changes and the results are compared with the existing approaches in the literature. It is shown that the proposed approach can handle any complex RMS configurations, solve the deadlock problem in an RMS, and detect and treat failures. Furthermore, is simpler in its structure.

INDEX TERMS Simulation, modeling, deadlock avoidance, colored Petri net, reconfigurable manufacturing system, neural network.

I. INTRODUCTION

The recent innovation in manufacturing is a reconfigurable manufacturing system (RMS). An RMS can be described as a series of discrete events, which characterizes a system. An RMS can modify its system structure, such as adding new machines, products, handling devices, and the rework of the process. To achieve these modifications, it requires a control program with a variety of features including quickness, validity, cost-effectiveness, and flexibility [1], [2]. In RMSs, when resources are shared, some operations may not be conducted because of deadlocks. Thus, deadlock control is necessary for RMSs. Moreover, in real world, the occurrence of resource faults may lead to new deadlocks. In general, faults are described as disturbances, failures, or mistakes that cause unbearable or unwanted resource behavior and thus cannot be ignored in the RMS. It is important to perform an early diagnosis and treatment of faults on machines and equipment to maintain efficiency and avoid performance degradations. It is necessary to design a deadlock prevention method in the RMS under unreliable resources, which can detect and treat faults.

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

Petri nets (PNs) are an excellent mathematical modeling tool widely used for deadlock control in automated manufacturing systems (AMSs) [1], [3]. PNs can be used to model the dynamic behaviors, e.g., concurrency, synchronization, causal dependence, sequencing, and conflict in automated manufacturing systems. Many policies have been developed in the literature based on Petri nets, focusing on three strategies: prevention of deadlocks, avoidance of deadlocks, and detection and recovery of deadlocks [4], [5]. Most of these policies assume that the resources in automated manufacturing systems are reliable [6]–[13], and others assume that they are unreliable [14]–[25]. Two analysis techniques in PNs are used to design deadlock control policies: reachability graph analysis [26]–[28] and structural analysis [3], [6]. In addition, three criteria are needed to design and evaluate the supervisor of AMS, which include structural complexity that leads to design a supervisor with a number of monitors [3], [7], [29], computational complexity that means a supervisor can be implemented on small and large sized systems [3], [30], and behavioral permissiveness that leads to improve the time performance (utilization, throughput, and throughput time) of the system [4], [25], [31].

Several studies have been proposed on deadlock control, faults diagnosis and treatment for unreliable resources in AMSs over the past several years. Two policies for one-unit resource allocation systems with unreliable resources are developed in [16]. Liu *et al.* propose deadlock control approaches for AMSs under unreliable resources using divide-and-conquer [18], a reachability graph partition approach [23], a max$'$-controlled siphon control method, and an elementary siphons approach [32]. Li *et al.* [33] propose a two-step deadlock control approach and a legal marking using an elementary siphon approach [34] for AMSs under unreliable resources. In [35], [36], a deadlock control policy based on a strict minimal siphon and colored Petri nets is proposed for AMSs with unreliable resources.

The study in [37] develops a multi-layered feedforward neural network method for defect diagnosis in industrial operations. The work in [38] uses a multi-layer perceptron neural network method to propose a continuous-time approach for a motor system for fault diagnosis and isolation. A distributed Petri net approach is used in [39] to develop a supervisor for fault detection in manufacturing systems operations. Bayesian networks and Petri diagnostic networks are integrated in [40] for defect diagnosis and treatment of malfunctions in automated machines. Miyagi and Riascos [41] integrate standard Petri nets and hierarchical networks; the integrated model is used to model and analyze fault-tolerant systems. Artificial neural networks are used in [42] for fault detection in technical systems, and a fuzzy neural network approach is proposed [43] for fault diagnosis in waste water treatment systems using online sensors. A deadlock control strategy based on neural networks and colored Petri nets is proposed in [44] for AMS for fault diagnosis and treatment under unreliable resources.

It is well-known that various deadlock control approaches have been developed with unreliable resources. However, the disadvantage of these approaches is that when a system is changed completely or a system is large, its size and complexity will also become large. Therefore, this paper aims to develop a four-step deadlock control policy for the RMS for fault diagnosis and treatment under unreliable resources. First, a CROTPN is designed for rapid and effective reconfiguration of the RMS without considering resource failures. Wu and Zhou have made significant contributions in deadlock avoidance using CROPN for AMSs, automated guided vehicles (AGV) systems, and robots [45]–[62]. Sufficient and necessary conditions are introduced at the second step for the liveness of the CROTPN to guarantee that the model is deadlock-free. The third step models the failures of all resources in the CROTPN model, where a single recovery subnet is developed and inserted into the CROTPN model at the second step to guarantee that the model is reliable. The fourth step integrates neural networks with the CROTPN for fault detection and treatment. The contributions of this paper are stated as follows.

1. A new approach is developed to solve the deadlock problem, detect and treat failures; compared with current studies, it can handle any dynamic changes in an RMS.
2. Sufficient and necessary conditions are introduced for the liveness of the CROTPN, which are more succinct than those in [1].
3. A single recovery subnet is developed to handle all resource failures in a CROTPN.
4. A simulation code is designed using the GPenSIM tool for modeling, validation, and performance comparison of the proposed CROTPN; the experimental results are compared with those of the current methods.

The rest of this study is structured as follows. The synthesis of the CROTPN is shown in Section II. Section III presents a deadlock avoidance policy for the CROTPN. The unreliable CROTPN is presented in Section IV. The integration of neural networks and the CROTPN for fault detection and treatment is presented in Section V. The application of the developed approaches is illustrated in Section VI. The conclusions, benefits, drawbacks, and future work of this study are described in Section VII.

II. DESIGN OF CROTPN

Most of Petri net models [5]–[13], [28], [31], [36], [63]–[70] do not undergo dynamic configurations, such as the addition of new machines, removal of old machines, addition of new products, processing rework, machine breakdowns, or change processing routes induced by the competitive global market. Therefore, a CROTPN is proposed to deal with dynamic configurations in an RMS. This section presents the formal definitions and properties of CROTPN.

A CROTPN is a directed bipartite graph with an initial state called the initial marking. It consists of two sorts of nodes: places, and transitions. Places are graphically drawn by circles and transitions by bars or boxes. There are directed arcs from a place to a transition or from a transition to a place, which labeled with their weights (positive integers). Each place can hold black dots (tokens), or a nonnegative integer representing their number. A marking assigns tokens to each place to represent a state of the modeled system.

Definition 1: An eight-tuple $N = (P, T, C, I, O, D, K, M_o)$ is said to be a CROTPN if

1. $P = \{p_o\} \cup \{p_r\} \cup P_R$, where p_o, p_r, and P_R are respectively an idle place (a load/unload station), a common material handling resource (transportation resources), and a finite set of resource places with $P_R = \cup_{i\in m}\{p_i\}$, where $m > 0$;
2. $T = \cup_{j\in n}\{t_j\}$ is a finite set of transitions with $n > 0$, $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
3. The sets $C(p_i)$ and $C(t_i)$ respectively correspond to the colors of place p_i and transition t_j, where
   a) $p_i \in P$, $u_i = |C(p_i)|$, $C(p_i) = \{a_{i1}, a_{i2}, \ldots, a_{iu}\}$
   b) $t_j \in T$, $v_j = |C(t_j)|$, $C(t_i) = \{b_{j1}, b_{j2}, \ldots, b_{jv}\}$
4. $I(p_i, t_j)(a_{ih}, b_{jk})$: $C(p_i) \times C(t_j) \to \mathbf{IN}$ denotes the input function of N and $O(p_i, t_j)(a_{ih}, b_{jk})$: $C(p_i) \times C(t_j) \to \mathbf{IN}$ denotes the output function of N, where $\mathbf{IN} = \{0, 1, 2, \ldots\}$;
5. D is a firing delay function that adds to each transition t the firing delay $D(t)$ with D: $T \to \mathbf{TS}$, where $\mathbf{TS} > 0$;
6. K: $P \to \mathbf{IN}$ denotes the capacity function that allocates the maximal number of tokens to each place $K(p_i)$;
7. M_o is the initial marking function that allocates the number of tokens to each place with color a_{ih} with M_o: $P \to \mathbf{IN}$, i.e., $M_o = (M_o(p_1, a_{1h}), M_o(p_2, a_{2h}), \ldots, M_o(p_m, a_{ih}))^T$, $a_{ih} \in C(p_i)$.

Definition 2: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let $F = I \cup O$. Let p_i and t_j be respectively a place and a transition node in N. Then ${}^{\cdot}p_i = \{t_j \in T |\ (t_j, p_i) \in F\}$ and $p_i^{\cdot} = \{t_j \in T |\ (p_i, t_j) \in F\}$ are a preset and postset of node p_i, respectively. ${}^{\cdot}t_j = \{p_i \in P |\ (p_i, t_j) \in F\}$ and $t_j^{\cdot} = \{p_i \in P |\ (t_j, p_i) \in F\}$ are the preset and postset of node t_j, respectively.

Definitions 3 and 4 introduce various CROTPN subclasses that fulfill particular structural conditions.

Definition 3: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. N is said to be an ordinary net if $I(p_i, t_j)(a_{ih}, b_{jk}) = 1, p_i \in P, t_j \in T, \forall(p_i, t_j) \in F$. It is called a weighted net if $\exists p_i \in P, \exists t_j \in T, \forall(p_i, t_j) \in F$ such that $I(p_i, t_j)(a_{ih}, b_{jk}) > 1$.

Definition 4: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. N is said to be self-loop free if $p_i \in P, t_j \in T, \forall(p_i, t_j) \in F, I(p_i, t_j)(a_{ih}, b_{jk}) > 0$ implies that $O(p_i, t_j)(a_{ih}, b_{jk}) = 0$. N contains a self-loop if $\exists p_i \in P, \exists t_j \in T, \forall(p_i, t_j) \in F, I(p_i, t_j)(a_{ih}, b_{jk}) > 0$ implies that $O(p_i, t_j)(a_{ih}, b_{jk}) > 0$.

The enabling and firing rules of transitions can be introduced based on the definitions of the input and output functions, colors, and markings.

Definition 5: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. At marking M, a transition t_j is said to be enabled with respect to color $b_{jk} \in C(t_j)$ if

$$\forall p_i \in P, \quad \forall p_i \in {}^{\cdot}t_j,\ a_{ih} \in C(p_i),\ M(p_i, a_{ih}) \geq I(p_i, t_j)(a_{ih}, b_{jk}) \tag{1}$$

and

$$\forall p_i \in P, \quad \forall p_i \in t_j^{\cdot},\ a_{ih} \in C(p_i),\ K(p_i) \geq M(p_i, a_{ih}) + O(p_i, t_j)(a_{ih}, b_{jk}) - I(p_i, t_j)(a_{ih}, b_{jk}) \tag{2}$$

Definition 5 indicates that t_j is enabled if conditions (1) and (2) are satisfied; then t_j is called process-enabled and resource-enabled.

At marking M, if a transition t_j is enabled, it can fire with respect to color b_{jk} and the marking changes from M to M' based on the following definition.

Definition 6: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. At marking M, an enabled transition t_j with respect to color $b_{jk} \in C(t_j)$ is represented by $M[t\rangle M'$; M' occurs from the firing of t_j and can be modified from M to M' in time interval $(\alpha_j, \alpha_j + D(t_j))$ as follows:

$$\forall p_i \in P, \quad a_{ih} \in C(p_i),\ M'(p_i, a_{ih}) = M(p_i, a_{ih}) + O(p_i, t_j)(a_{ih}, b_{jk}) - I(p_i, t_j)(a_{ih}, b_{jk}) \tag{3}$$

where transition t_j starts firing at time α_j and the time delay for firing t_j is $D(t_j)$.

The incidence matrix describes the dynamic behaviour of nets, which determines all possible interconnections between places and transitions in a net. The incidence matrix for CROTPN is defined as follows.

Definition 7: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let $[N]$ be the incidence matrix of N, where $[N]$ is a $\sum_{i=1}^{m} u_i \times \sum_{j=1}^{n} v_j$ matrix with $[N_{ij}] = O(p_i, t_j)(a_{ih}, b_{jk}) - I(p_i, t_j)(a_{ih}, b_{jk})$.

Definition 8: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let A be a finite set of colors with cardinality $|A| = l$, $A \neq \emptyset$. Let Y be a set of places function with respect to A, represented on p_i as $Y(p_i)$: $A\times C(p_i) \to \mathbf{IZ}$, where $\mathbf{IZ} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

Definition 9: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let Ψ be a weighted matrix with $\Psi = [\Psi_1, \Psi_2, \ldots, \Psi_m]^T$ that indicates a set of places, where Ψ_i is a matrix with $l \times u_i$ integer dimensions. Ψ is said to be a place invariant of N if

$$[N]^T \Psi = \mathbf{0} \tag{4}$$

Definition 10: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let $M(C) = (M(p_1, a_{11}), M(p_1, a_{12}), \ldots, M(p_1, a_{1\ u1}), M(p_2, a_{21}), \ldots, M(p_2, a_{2\ u2}), \ldots, M(p_m, a_{m1}), \ldots, M(p_m, a_{mum}))^T$. If Ψ is a place invariant of a CROTPN, then we have

$$\forall M \in R(N, M_o), \quad M(C)^T Y = M_0(C)^T Y \tag{5}$$

where $R(N, M)$ denotes the set of reachable markings from M in N.

The behavioral properties in a CROTPN are essential in the analysis and control of a system. Some of the most important behavioral properties are introduced in the following definitions, which are conservativeness, boundedness, safeness, and reversibility.

Definition 11: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let x be a positive integer vector with $x = [x_1, x_2, \ldots, x_m]$. N is said to be conservative if

$$\forall M \in R(N, M_o), \quad M^T x = M_0^T x \tag{6}$$

Definition 12: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. N is called a bounded net if $\forall p_i \in P$, $a_{ih} \in C(p_i)$, $\forall M \in R(N, M_o)$, $M(p_i, a_{ij}) \leq q$, $q \in \{1, 2, 3, \ldots\}$. N is called a safe net if $\forall p_i \in P$, $a_{ih} \in C(p_i)$, $\forall M \in R(N, M_o)$, $M(p_i, a_{ij}) \leq 1$. N is called q –safe if it is q-bounded.

Definition 13: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. If M_o is reachable from a marking $M' \in R(N, M_o)$, then a marking M_o is said to be reversible.

Finally, the processing routes of parts in a CROTPN can be described as the following definition.

Definition 14: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$, where $PR = \{PR_1, PR_2, PR_3, \ldots, PR_\pi\}$ indicates all feasible processing routes for all part types π, $\pi = \{1, 2, 3, \ldots\}$. PR_i is $R_o \rightarrow R_{i1} \rightarrow R_{i2} \rightarrow \ldots\ldots \rightarrow R_{i\varepsilon i} \rightarrow R_o$, $\varepsilon_i = \{1, 2, 3, \ldots\}$. R_o indicates an infinite load/unload area of N, and $R_i (i \neq 0)$ indicates a machine (resource). The processing sequence starts at R_o and finishes at R_o. In the existence of more than one processing sequence for each part such that $R_i \rightarrow R_{j(i \neq j)}$, there exists a sequence from R_i to R_j. If the part shifts from R_i to R_j, a part handling device (called a transportation resource) such as an AGV or a robot, is needed to move the part.

Based on Definitions 1–14, the developed CROTPN for modeling processing routes of the system is constructed in Algorithm 1.

III. DEADLOCK AVOIDANCE POLICY FOR CROTPN

A CROTPN includes many circuits owing to its high connectedness. A production process circuit (PPC) is a special circuit in a CROTPN and plays a vital role in the liveness of the CROTPN. Because of the routing complexity of an RMS, a CROTPN may contain multiple PPCs, but only some of them can be found in a CROTPN.

Definition 15: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let PPCs be circuits (which do not include place p_o) in a CROTPN, expressed as PPCs $= \{e_1, e_2, \ldots, e_k\}$, $k = \{1, 2, 3, \ldots\}$. A PPC e_k is called an elementary circuit if it moves from a node z, through several nodes, back to the starting node z, and no node is repeated.

If a PPC e_k does not fulfill the condition given in Definition 15, then the PPC e_k is said to be a nonelementary.

The number of PPC e_k places must be equal to that of transitions on e_k in a CROTPN, and the transition input place for e_k must be on e_k.

Algorithm 1 Construction of a CROTPN

Input: A Part type $i \in \pi$, and its processing route PR_i.
Output: A net CROTPN.
Initialization: $P = \{p_o, p_r\}$, $T = \emptyset$, $F = \emptyset$, $M_o = \emptyset$, $C = \emptyset$, $\mu = \{R_{io}\}$, $i = 0$, and $j = 0$.

```
for (1, |π|, i++), choose PR_i ∈ PR, do
    for (1, |ε_i|, j++), do
        For R_i(j−1), add a place p_x;
        For R_ij, add a place p_y;
        Add a transition t_xyi;
        Add arcs and weights (p_x, t_xyi) and (t_xyi, p_y);
        P := P ∪ {p_x, p_y} and T := T ∪ {t_xyi};
        F := F ∪ ((P × T) ∪ (T × P));
        μ := μ ∪ {R_ij};
        if t_xyi needs p_r to transfer part type i, then
        Add arcs and weights (p_r, t_xyi) and (t_xyi, p_r);
        F := F ∪ ((P × T) ∪ (T × P));
        end if
    end for
        For R_i0i, add a place p_x;
        Add a transition t_x0i;
        Add arcs and weights (p_x, t_x0i) and
         (t_x0i, p_o);
        T := T ∪ {t_x0i};
        F := F ∪ ((P × T) ∪ (T × P));
          if t_x0i needs p_r to transfer part type i,
          then
             Add arcs (p_r, t_x0i) and (t_x0i, p_r);
             F := F ∪ ((P × T) ∪ (T × P));
          end if
end for
 /* Construct the M_o* /
        M_o(p_o) = {c_p1, c_p2, ... c_pn};
        M_o(p_r) = {c_t1, c_t2, ... c_tk};
        M_o(p) = 0, ∀p ∈ P_R;
        M_o := M_o ∪ M_o(p_o) ∪ M_o(p_r) ∪ M_o(p);
        C := C(p_o) ∪ C(p_r).
Output a net CROTPN.
End
```

Definition 16: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let $P(e_k)$ and $T(e_k)$ be respectively the sets of places and transitions in e_k such that $|P(e_k)| = |T(e_k)|$, $p_i \in \cdot t_j$, and $\cdot t_j \in P(e_k)$. Let $M(p_i, e_k)$ be the number of tokens in place p_i that enables t_j in e_k. If t_j is fired and the tokens leave e_k, then the tokens in e_k are called the leaving tokens of e_k. If t_j is fired and the tokens do not leave e_k, then the tokens in e_k are called the cycling tokens of e_k, expressed as

$$p_i \in P(e_i), \quad M(e_i) = \Sigma M(p_i, e). \tag{7}$$

The interaction of PPCs in CROTPN complicates the liveness problem of the net. The following definitions discuss interactive PPC subnets.

Definition 17: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. A circuit e_k^n is said to be interactive, consisting of n PPCs, if it is strongly connected and its places and transitions are shared with at least another PPC. Let $P(e_k^n)$ and $T(e_k^n)$ be respectively the sets of places and transitions in e_k^n such that $T_i = \{t \in p_i{\cdot} \cap T(e_k^n)\}$ and $p_i \in P(e_k^n)$. Let $M(p_i, e_k^n)$ be the number of tokens in place p_i that enables t_j in e_k^n. If $t_j \in T_i$ is fired and the tokens leave e_k^n, then the tokens in e_k^n are called the leaving tokens of e_k^n. If $t_j \in T_i$ is fired and the tokens do not leave e_k^n, then the tokens in e_k^n are called the cycling tokens of e_k^n, expressed as

$$p_i \in P(e_k^n), \quad M(e_k^n) = \Sigma M(p_i, e_k^n) \tag{8}$$

Definition 18: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. At marking M, a PPC e_k has no free space in the places (called full) if

$$p_i \in P(e_k), \quad \Sigma M(p_i) = \Sigma K(p_i) = K(e_k) \tag{9}$$

Theorem 1: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. The necessary condition for N to be deadlocked (not live) is that there exists at least one PPC e such that

$$M_o(p_o) \geq K(e_k). \tag{10}$$

Proof: Proved in [50].

Before addressing deadlock-free conditions and control policy in a CROTPN, we introduce some necessary definitions and notation.

Definition 19: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. A transition t_j is called a controlled transition if the t_j enabling conditions (1) and (2) in Definition 5 are satisfied. If at least one transition is controlled in the N, then it is called a controlled transition. A PPC e_k in the N is enabled when it is process– and resource–enabled. A PPC e_k is called a live transition if for each $t_j \in T(e_k)$, t_j is live.

Consequently, the CROPN's control policy is limited. It decides whether any controlled transition may fire by monitoring the net state, even if both process and resource are satisfied. If a controlled transition can fire based on a control policy, we claim that this policy allows control.

Definition 20: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let e_k^n be an interactive subnet in the N. A transition t_j is said to be an input transition of e_k^n if $t_j^{\cdot} \in P(e_k^n)$ and $t_j \notin T(e_k^n)$. A transition t_j is called an output transition of e_k^n if ${}^{\cdot}t_j \in P(e_k^n)$ and $t_j \notin T(e_k^n)$. Let $T_I(e_k^n)$ and $T_O(e_k^n)$ be respectively the sets of input transitions and output transitions of e_k^n.

Definition 21: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let the number of current spaces and free spaces in PPC e_k be respectively $S'(e_k)$ and $S(e_k)$ at marking M, expressed by

$$S(e_i) = \sum\nolimits_{p_j \in P(\boldsymbol{e_k})} (K(p_j) - M(p_j)) \tag{11}$$

$$S'(e_i) = K(e_k) - M(e_k) \tag{12}$$

Theorem 2: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let $R_{DF}(N, M_o)$ denote the set of reachable markings under control in the N. For any marking $M \in R_{DF}(N, M_o)$, a PPC e_k is live if

$$S'(e_k) \geq 1. \tag{13}$$

Proof: Proved in [50].

Theorem 3: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let $\eta(e_k^n, M)$ denote the enabled PPCs in e_k^n at marking M. For any marking $M \in R_{DF}(N, M_o)$ reachable from M_o, a PPC e_k^n is live if

$$S'(e_k) \geq 1, \quad \text{for any } e_k, \tag{14}$$

and

$$\eta(e_k^n, M) \geq 1 \tag{15}$$

Proof: Proved in [50].

Theorem 4: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. If there is no PPC in the N, then it is always live.

Proof: Proved in [50].

Definition 22: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. Let a transition t_j be an input transition of a number of PPCs. Let $V_{en}(t_j)$ and T_d be respectively the set of these PPCs and the set of transitions in the N such that if $t_j \in T_d$, then $V_{en}(t_j) \neq \emptyset$.

Theorem 5: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o, PR)$. At marking M, the e_k^n is live if (1) each $t_j \in T_I(e_k^n)$ is controlled; (2) before t_j fires, for any $e_k \in V_{en}(t_j)$, $S'(e_k) \geq 2$; and (3) after t_j fires, the marking in the N is updated from M to M' with $\eta(e_k^n, M') \geq 1$.

Proof: Proved in [50].

Theorems 1–5 in [50] present the necessary and sufficient deadlock-free conditions for the CROTPN. Based on Definitions 15–22 and Theorems 1–5, the deadlock avoidance algorithm of the developed CROTPN is constructed in Algorithm 2.

IV. DESIGN OF UNRELIABLE CROTPN

A resource failure in an RAMS is a matter of temporal uncertainty. If a resource fails in an unreliable place p_i, we attempt to add a subnet that is capable of removing a token from p_i and repairing the failed resource. Additionally, this subnet will return a token to the unreliable place after the resource is repaired. The resource can then be reused. This subnet is called a recovery subnet. This section presents the formal definitions that are used to construct the recovery subnets for all failures in a CROTPN.

Definition 23: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. Let $r_u \in P_R$ be an unreliable resource in N. Let N_{RNi} be a recovery subnet of r_u with $N_{RNi} = (\{p_i, p_{combined}\}, \{t_{fi}, t_{ri}\}, F_{rni}, c_{rni})$, where $p_i \in P_R$ and $p_{combined}$, t_{fi}, t_{ri} represent respectivelythe common recovery place of p_i, the failure transition, and the recovery transition. $F_{rni} = \{(p_i, t_{fi}), (t_{fi}, p_{combined}), (p_{combined}, t_{ri}), (t_{ri}, p_i)\}$. c_{rni} is the color that maps $p_i \in P$ into colors $c_{rni} \in C$. (N_{RNi}, M_{RNio}) is called a marked recovery subnet, where $M_{RNio}(p_i) \geq 0$ and $M_{RNio}(p_{combined}) = 0$.

Algorithm 2 Deadlock Avoidance Policy of the CROTPN

Input: A net CROTPN.
Output: Controlled net CROTPN.
Initialization: $k = 0$ and $j = 0$.
Compute the set of PPCs $= \{e_1, e_2, \ldots, e_k\}$ for CROTPN.
Compute the set of $R(N, M_o)$ for CROTPN.
for *(1, | PPCs |, k++),* ***do***
if e_k is not an interactive subnet, ***then***
Check the liveness of e_k;
for *(0, |R (N, M_o)|, j++),* ***do***
$K(e_k) = \Sigma K(p, e_k), p \in P(e_k)$;
$M_j(e_k) = \Sigma M_j(p, e_k), p \in P(e_k)$;
$S'(e_k) = K(e_k)$- $M_j(e_k)$;
if $S'(e_k) \geq$, ***then*** /* By using Theorems 1 and 2. */
e_k in a CROTPN is live;
$R_{DF}(N, M_o) = R(N, M_o)$;
else if
e_k in a CROTPN is not live;
end if
else if /* e_k is an interactive subnet with n PPCs */.
Check the liveness of e_k^n;
for *(0, |R(N, M_o)|, j++),* ***do***
$K(e_k^n) = \Sigma K(p, e_k^n), p \in P(e_k^n)$;
$M_j(e_k) = \Sigma M_j(p, e_k), p \in P(e_k^n)$;
$S'(e_k^n) = K(e_k^n)$- $M_j(e_k^n)$;
if $S'(e_k) \geq 1$ an $\eta(e_k^n, M_j) \geq$, ***then*** /* By using Theorems 3-5. */
e_k^n in a CROTPN is live;
$R_{DF}(N, M_o) = R(N, M_o)$;
else if
e_k^n in a CROTPN is not live;
end if
Output a controlled net CROTPN.
End

Definition 24: Let N be a CROTPN with $N = (P, T, C, I, O, D, K, M_o)$. For all $r_u \in P_R$, designing a common recovery subnet results in an unreliable CROTPN, expressed by $(N_U, M_{Uo}) = (N, M_o) \parallel (N_{RNi}, M_{RNio})$ that is the composition of (N, M_o) and (N_{RNi}, M_{RNio}), where $\parallel$ means the net composition.

Definition 25: Let (N_U, M_{Uo}) be an unreliable CROTPN with $N_U = (P_U, T_U, C_U, I_U, O_U, D_U, K_U, M_{Uo})$, and $R_U(N_U, M_{Uo})$ be its reachable graph, where $P_U = P \cup \{p_{combined}\}$, $T_U = T \cup T_F \cup T_R$. T_F and T_R represent respectivelythe sets of failure transitions and the recovery transitions of (N_U, M_{Uo}) with $T_F = \cup_{i\in \mathrm{NA}}\{t_{fi}\}$, $T_R = \cup_{i\in \mathrm{NA}}\{t_{ri}\}$, and $\mathbf{NA} = \{i|p_i \in P_R\}$. $C_U = C \cup C_F$, $C_F = \cup_{i\in \mathrm{NA}}\{c_{rni}\}$. $I_U(p_i, t_j)$: $C_U(p_i) \times C_U(t_j) \to \mathbf{IN}$, $O_U(p_i, t_j)$: $C_U(p_i) \times C_U(t_j) \to \mathbf{IN}$, D_U: $T_U \to \mathbf{TS}$, K_U: $P_U \to \mathbf{IN}$, and M_{Uo}: $P_U \to \mathbf{IN}$ is an initial marking of N_U.

Based on Definitions 23–25, an algorithm for an unreliable CROTPN is constructed in Algorithm 3.

Algorithm 3 Unreliable CROTPN Construction

Input: A net CROTPN.
Output: An unreliable CROTPN (N_U, M_{Uo}).
Initialization: $P_U = P \cup \{p_{combined}\}$, $T_U = T$, $F_U = F$, $C_U = C$, $M_{Uo} = M_o$, and $i = 0$.
for *(1, |P_R|, i++),* ***do***
Add a transition t_{fi};
Define a color c_{rni} for t_{fi};
Add a transition t_{ri};
Add an arc (p_i, t_{fi});
Add an arc $(t_{fi}, p_{combined})$;
Add an arc $(p_{combined}, t_{ri})$;
Add an arc (t_{ri}, p_i);
$T_U := T_U \cup \{t_{fi}, t_{ri}\}$;
$F_U := F_U \cup ((P_U \times T_U) \cup (T_U \times P_U))$;
$C_U := C_U \cup \{C_{rni}\}$.
end for
/* Define the M_{Uo}^*/
$M_{Uo} := M_o \cup M_{Uo}(p_{combined})$.
Output an unreliable CROTPN net.
End

V. NEURAL NETWORK AND CROTPN FOR FAULT DETECTION AND TREATMENT

Recently, neural networks (NNs) have gained popularity since they can learn complex functions. Parallel and distributed processing systems comprising a large number of simple and highly-connected processors can be seen as neural networks. These networks can be trained offline for complicated mapping, for example to determine different faults and can then be effectively used in the online environment. This section presents the formal definitions that are used to construct the CROTPN model based on neural networks for fault detection and treatment.

Definition 26: $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_\zeta, Y_\zeta, W_{NP}, M_{NPo})$ is said to be a neural Petri net (NPN) if

1. P_{NP} is a set of places that represent the input neurons p_{xi} and outputs pattern p_{yj} of the neural network with $P_{NP} = \cup_{i\in\theta} \{p_{xi}\} \cup (\cup_{j\in\alpha} \{p_{yj}\})$, $\theta, \alpha > 0$;
2. T_{NP} is a set of transitions that represent fault detection t_{di} and fault treatment t_{ti} with $T_{NP} = \cup_{i\in\beta}\{t_{di}\} \cup (\cup_{j\in\gamma}\{t_{tj}\})$ and $\beta, \gamma > 0$;
3. F_N is the input and output function with $F_N \subseteq (P_{NP} \times T_{NP}) \cup (T_{NP} \times P_{NP})$;
4. X_ζ denotes the set of inputs neuron pattern k that represents the input factor of the neural model with $X_\zeta = \cup_{i\in\theta}\{x_i^\zeta\}$, where each x_i is mapped to the corresponding p_{xi} and $\zeta = 1, 2, 3 \ldots$;
5. Y_ζ denotes the set of outputs of pattern ζ that represents the output factor of the neural model with $Y_\zeta = \cup_{j\in\alpha}\{y_j^\zeta\}$, where each y_j is mapped to the corresponding p_{yj};
6. $W_{NP} \to [0, 1]$ denotes the set of the connectivity matrix from p_{xi} to t_{di};

7. M_{NPo}: $P_{NP} \rightarrow$ **IN** denotes the initial marking of N_{NP}.

The major difference between the standard NN and the NPNs is that the Petri layer and transition layer proposed in NPNs represent the configuration of the NPNs model of the fault detection and treatment. The Petri layer input x_i^{ζ} is the input of the NPNs and the output of the each node in this Petri layer is tokens with acquisition systems that collect signals from input sensors and connectivity matrix W_{NP}, which is defined as follows:

Definition 27: Let $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NPo})$ be a neural Petri net model. The connectivity matrix W_{NP} of N_{NP} can be represented as:

$$W_{NP} = \begin{bmatrix} \sum w_{ij} = 1 & \begin{matrix} j \in \beta \text{ and } i|p_{xi} \\ \in \ ^{\cdot}t_{dj} \end{matrix} \\ w_{ij} = 0 & \text{otherwise} \end{bmatrix} \tag{16}$$

Definition 28: Let $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NPo})$ be a neural Petri net model. Let Z_j be the input of all the fault detection neuron layers. If t_{dj} is enabled and fired, then the input Z_j is generated and represented as:

$$Z_j = \begin{bmatrix} \sum w_{ij} x_i^{\zeta} & \begin{matrix} j \in \beta \text{ and } i| \\ x_i \in X_{\zeta} \end{matrix} \\ 0 & \text{otherwise} \end{bmatrix} \tag{17}$$

The input of the transition layer is the output of the Petri layer and is connected to the neural network middle layer. This layer is designed to generate tokens using competition laws as follows:

Definition 29: Let $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NPo})$ be a neural Petri net model. The input $Z_j \in Y_{\zeta}$ is called the winner if it has the largest value compared with others' input values and its output value y_j is stated as 1; otherwise, the other output values are stated as 0, denoted as

$$Y_{\zeta} = \begin{bmatrix} y_j = 1 & \begin{matrix} Z_j > Z_i \\ i > 0 \\ i \neq j \end{matrix} \\ y_i = 0 & i \neq j \end{bmatrix} \tag{18}$$

We define the update law on the synaptic weight of the winner neuron w_{ij}, to guarantee the NPNs precisely online estimation as follows:

Definition 30: Let $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NPo})$ be a neural Petri net model. Let the winner j^{th} output neuron be y_j. Thus, the synaptic weight of the winner neuron's $w_{ij} \in W_{NP}$ can be formulated as.

$$w_{ij} = \begin{bmatrix} w_{ij} + \Delta w_{ij} & \begin{matrix} i \in \theta \\ j \in \beta \end{matrix} \\ w_{ij} & \text{Otherwise} \end{bmatrix} \tag{19}$$

$$\Delta w_{ij} = \lambda \left(\frac{x_i^{\zeta}}{\delta} w_{ij} \right) \quad \begin{matrix} i \in \theta \\ j \in \beta \end{matrix} \tag{20}$$

where δ denotes the number of items, which are equal to 1 in the input learning pattern X_{ζ} and $\lambda \rightarrow [0, 1]$ denotes a learning rate.

Finally, the neural unreliable CROTPN can be described as the following definition.

Definition 31: Let (N_U, M_{Uo}) be an unreliable CROTPN with $N_U = (P_U, T_U, C_U, F_U, W_U, D_U, K_U, M_{Uo})$ and let $N_{NP} = (P_{NP}, T_{NP}, F_{NP}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NPo})$ be a neural Petri net model. We call (N_{NU}, M_{NUo}) a neural unreliable CROTPN, expressed as $(N_{NU}, M_{NUo}) = (N_U, M_{Uo}) \parallel (N_{NP}, M_{NPo})$ that is the composition of (N_U, M_{Uo}) and (N_{NP}, M_{NPo}), where $N_{NU} = (P_{NU}, T_{NU}, C_{NU}, I_{NU}, O_{NU}, D_{NU}, K_{NU}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NUo})$, and

1. $P_{NU} = P_U \cup P_{NP}$;
2. $T_{NU} = T_U \cup T_{NP}$;
3. $C_{NU} = C_U$;
4. $I_{NU}(p_i, t_j)$: $C_{NU}(p_i) \times C_{NU}(t_j) \rightarrow$ **IN**;
5. $O_{NU}(p_i, t_j)$: $C_{NU}(p_i) \times C_{NU}(t_j) \rightarrow$ **IN**;
6. D_{NU}: $T_{NU} \rightarrow$ **TS**;
7. K_{NU}: $P_{NU} \rightarrow$ **IN**;
8. M_{NUo}: $P_{NU} \rightarrow$ **IN** is the initial marking of N_{NU}.

Based on Definitions 36–31, the developed training and solution of a neural unreliable CROTPN algorithm is constructed in Algorithm 4.

Algorithm 4 Fault Type of a Neural Unreliable CROTPN

Input: A neural unreliable CROTPN N_{NU} with $N_{NU} = (P_{NU}, T_{NU}, C_{NU}, F_{NU}, W_{NU}, D_{NU}, K_{NU}, X_{\zeta}, Y_{\zeta}, W_{NP}, M_{NUo})$.
Output: The fault type Y_{ζ}
Initialization: $w_{ij} \rightarrow [0, 1]$, X_{ζ}, μ (target weight), $r = 0$, $i = 0, j = 0$, and $k = 0$.

for $(1, |T_F|, r{+}{+})$, ***do***
  if t_{fl} fires, ***then***
    while $w_{ij} < \theta$, **do**
      for $(1, |\zeta|, k++)$, ***do***;
        for $(1, |\theta|, i++)$, ***do***;
          for $(1, |\beta|, j++)$, ***do***;
            Select a pattern X_{ζ} from the input patterns ζ;
            Compute the input Z_j of all the neurons;
            Compute the winner output value y_j and Enable the winner transition t_{dj};
            Update the weight w_{ij} of the winner neuron.
          end for
        end for
      end for
    end while
  else if
    Break.
  end if
end for
Output a fault type Y_{ζ}.
End

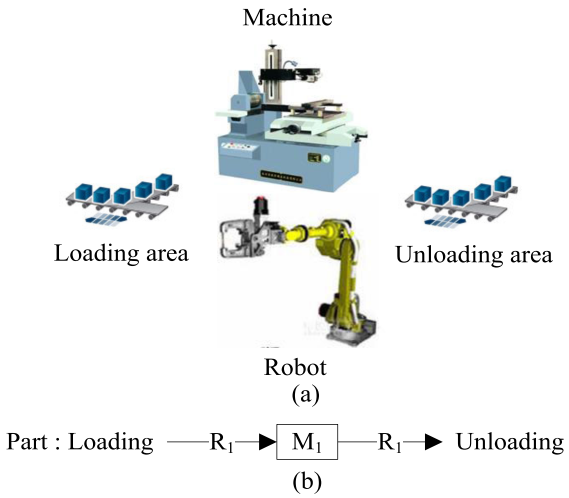

FIGURE 1. (a) Example of AMS and (b) The process route.

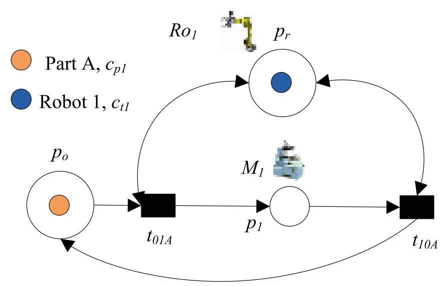

FIGURE 2. CROTPN of the system shown in Figure 1.

VI. NUMERICAL EXAMPLE

A. ALGORITHM 1 APPLICATION

Consider the AMS illustrated in Figure 1(a) to demonstrate the procedures of a CROTPN synthesis. The operation route of the system is presented in Figure 1(b). Figure 2 displays the transportation and operation resources of the system after implementing Algorithm 1, where resource place p_1 represents machine 1 (M1), the operation sequence of part A is $p_o \rightarrow t_{01A} \rightarrow p_1 \rightarrow t_{10A} \rightarrow p_0$, and a transportation place p_r represents robot 1 (Ro1). To build the model of processes of transportations of part types, a place p_r is needed respectively for loading and unloading part A to and from p_1 by transitions t_{01A} and t_{10A}. Thus, arcs (p_r, t_{01A}), (t_{01A}, p_r), (p_r, t_{10A}), and (t_{10A}, p_r) are added into the CROTPN model. The CROTPN initial marking is defined as $M_o(p_o) = \{c_{p1}\}, M_o(p_r) = \{c_{t1}\}$, and $M_o(p_1) = 0$, where c_{p1} and c_{t1} denote respectively part type A and robot 1 in the system.

Based on Definitions 5 and 6, the behavior of the CROTPN illustrated in Figure 2 is explained as follows. When transition t_{01A} fires, it selects respectively a token with color c_{p1} and color c_{t1} from input places p_o and p_r. If t_{01A} fires, it adds respectively a token c_{p1} and color c_{t1} to output places p_1 and p_r. Finally, when t_{10A} fires, it selects respectively a token c_{p1} and color c_{t1} from input places p_1 and p_r. If t_{10A} fires, it adds respectively a token with color c_{p1} and color c_{t1} to output places p_o and p_r.

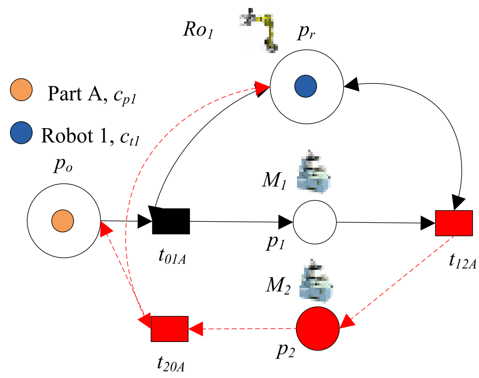

FIGURE 3. CROTPN after adding a new machine for the system shown in Figure 2.

To design and model the dynamic changes of the system shown in Figure 1(a) by using CROTPN synthesis, we assume that a system undergoes configurations including:

1. Adding a new machine;
2. Adding a new product;
3. Rework;
4. Adding a new transportation resource.

The first reconfiguration adds a new machine M2 to process part A after machine 1 in Figure 1(a), and robot 1 is required to load/unload part A to/from machine 2. To do this, consider the following steps to build the changed system by using the CROTPN synthesis:

1) Update the operation sequence of part A as: $p_o \rightarrow t_{01A} \rightarrow p_1 \rightarrow t_{12A} \rightarrow p_2 \rightarrow t_{20A} \rightarrow p_0$;
2) Insert the new transitions t_{12A} and t_{20A} to the model;
3) Draw arcs (p_r, t_{12A}), (t_{12A}, p_r), (p_r, t_{20A}), and (t_{20A}, p_r).

The new reconfigured CROTPN model is shown in Figure 3.

The second dynamic change adds a new product part B to the model shown in Figure 3. Part B requires to be processed in machine 1 and robot 1. Ro1 is required to load/unload part B to/from machine 1. To build the newly added product, we apply the following steps to construct the changed system by using the CROTPN synthesis:

1) Add a new process sequence: $p_o \rightarrow t_{01B} \rightarrow p_1 \rightarrow t_{10B} \rightarrow p_0$;
2) Insert the new transitions t_{01B} and t_{10B};
3) Draw arcs (p_r, t_{01B}), (t_{01B}, p_r), (p_r, t_{10B}), and (t_{10B}, p_r);
4) Add the color c_{p2} as an initial token into p_o that represents part B in the system.

The new reconfigured CROTPN model is shown in Figure 4.

The third control specification is to add a rework sequence to part B by inserting an inspection machine 3 after machine 1 in Figure 4. If Part B is successfully produced after machine 1 operation, the system will continue according to its initial route. Otherwise, if defects occur in part B, then rework is necessary. To do this, consider the following steps to build the changed system by using the CROTPN synthesis:

1) Update the operation sequence of part B as
   a. $p_o \rightarrow t_{01B} \rightarrow p_1 \rightarrow t_{13B} \rightarrow p_3 \rightarrow t_{30B} \rightarrow p_0$;

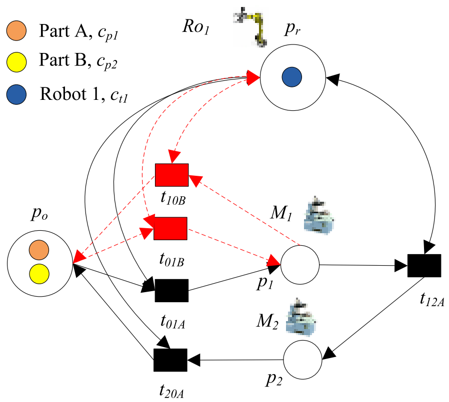

FIGURE 4. The CROTPN model after adding a new product for the system shown in Figure 3.

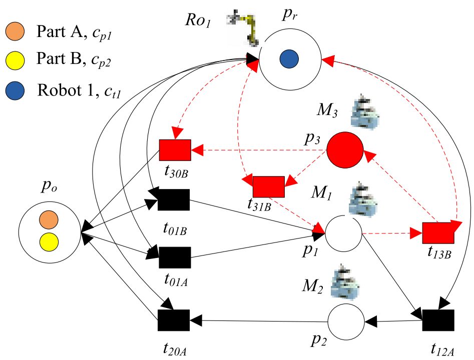

FIGURE 5. CROTPN after adding a rework for the system shown in Figure 4.

b. $p_o \rightarrow t_{01B} \rightarrow p_1 \rightarrow t_{13B} \rightarrow p_3 \rightarrow t_{31B} \rightarrow p_1 \rightarrow t_{13B} \rightarrow p_3 \rightarrow t_{30B} \rightarrow p_0$

2) Insert the new transitions t_{13B}, t_{31B}, and t_{30B}.
3) Draw arcs (p_r, t_{13B}), (t_{13B}, p_r), (p_r, t_{31B}), (t_{31B}, p_r), (p_r, t_{30B}), (t_{30B}, p_r).

The new reconfigured CROTPN model is shown in Figure 5.

Finally, suppose that a robot Ro2 (transportation resource) is inserted into the CROTPN shown in Figure 5 to load/unload part A to/from machine 1 and machine 2. To build the newly added robot Ro2, we apply the following steps to construct the changed system by using the CROTPN synthesis:

1) Add the initial token with color c_{t2} into place pr, indicating that robot 2 in the system.
2) Update the transitions t_{01A}, t_{12A}, and t_{20A};

The new reconfigured CROTPN model is shown in Figure 6.

B. ALGORITHM 2 APPLICATION

Let us reconsider the CROTPN of the changed model presented in Figure 6 to display the liveness of the CROTPN by using Algorithm 2. It has one PPC: $e_1 = \{p_1, t_{13B}, p_3, t_{31B}\}$. The CROTPN reachability graph in Figure 6 is illustrated in Figure 7. We assume that all places except p_r and p_o have a capacity of one with a buffer and a machine space. According to Theorem 2, the available spaces in e_1 are calculated as shown in Table 1. We can see in Table 1 that for any marking, the condition in Theorem 2 is satisfied. Therefore, the deadlock in the reconfigured CROTPN shown in Figure 6 can be avoided when the condition illustrated in Theorem 2 is applied. The GPenSIM tool [3], [71] is employed to verify and validate Algorithms 1 and 2 and build a CROTPN model code. The proposed code is implemented on MATLAB R2015a. The simulation was done for 480 min. The time performance includes the total throughput time, total throughput, and utilization of machines and robots. The experimental results are compared with those in [3], [7], [30], [72]. Reconsider the CROTPN shown in Figure 3. Table 2 illustrates the MATLAB simulation results with regard to the above time performance criteria. It shows that Algorithm 1 can achieve less throughput time, greater throughput, and better utilization than the other methods, as presented in Figure 8. Thus, Algorithm 1 is accurate.

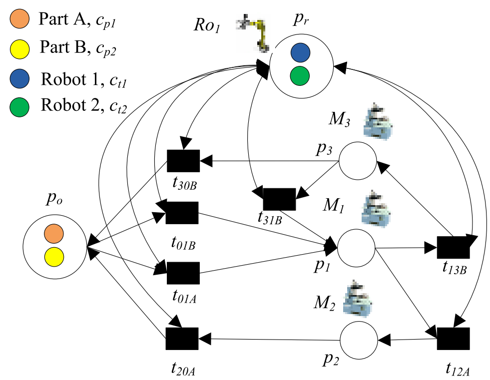

FIGURE 6. CROTPN model after adding a new transportation resource to the system shown in Figure 5.

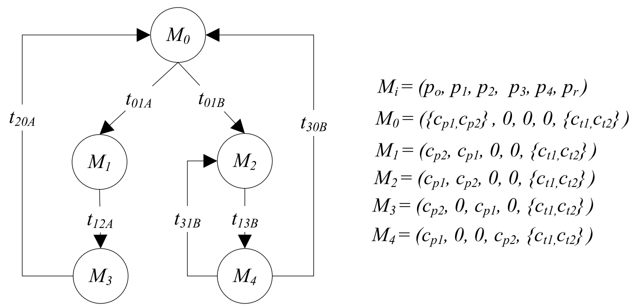

FIGURE 7. Reachable markings of the system shown in Figure 6.

TABLE 1. The available spaces in e_1 for CROTPN shown in Figure 6.

Marking	$K(e_1) = \Sigma K(p_i, e_1)$	$M(e_1) = \Sigma M(p_i, e_1)$	$S'(e_1) = K(e_1) - M(e_1)$	If $S'(e_1) \geq 1$
M_o	2	0	2	Yes
M_1	2	1	1	Yes
M_2	2	1	1	Yes
M_3	2	0	2	Yes
M_4	2	1	1	Yes

TABLE 2. **The time performance of Algorithm 1 compared with the existing approaches for the model shown in Figure 3.**

Parameters	Ref. [72]	Ref. [7]	Ref. [3]	Ref. [30]	Algorithm 1
Throughput	34	34	34	34	36
Throughput time	14.12	14.12	14.12	14.12	13.33
M1 utilization	29.05	29.05	29.60	29.05	30.83
M2 utilization	29.61	29.61	30.50	29.61	35.00
R1 utilization	48.04	48.04	47.56	48.04	42.08

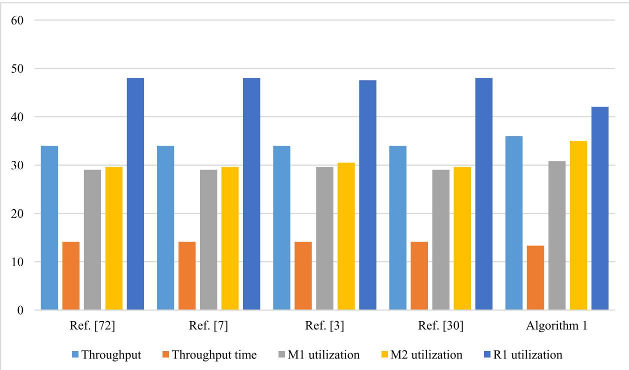

FIGURE 8. **The time performance of Algorithm 1 compared with the current approaches for the net shown in Figure 3.**

C. ALGORITHM 3 APPLICATION

To illustrate the common recovery subnet of an unreliable CROTPN by using Algorithm 3, consider the CROTPN model presented in Figure 6. We have three unreliable machines: p_1, p_2, and p_3. Adding the recovery subnet by using Algorithm 3 results in an unreliable CROTPN, as shown in Figure 9, where $\mathbf{NA} = \{1, 2, 3\}$, $T_F = \{t_{f1}, t_{f2}, t_{f3}\}$, and $T_R = \{t_{r1}, t_{r2}, t_{r3}\}$, and $C_F = \{c_{rn1}, c_{rn2}, c_{rn3}\}$. The behavior of the unreliable CROTPN illustrated in Figure 9 is explained as follows. When an unreliable machine 1 breaks down in p_1, the token in p_1 moves into $p_{combined}$ by firing t_{f1}. If t_{f1} fires, it generates a token c_{rn1} and deposits it into a place $p_{combined}$. If the mean time of performing maintenance on machine 1 is elapsed, the token c_{rn1} in $p_{combined}$ moves into p_1 by firing t_{r1}. If transition t_{r1} fires, it takes one token c_{rn1} from $p_{combined}$ and deposits it into p_1, denoting that a machine 1 recovery maintenance is finished.

If an unreliable machine 2 breaks down in p_2, then the token in p_2 moves into $p_{combined}$ by firing t_{f2}. If t_{f2} fires, it creates a token c_{rn2} and deposits it into a place $p_{combined}$. If the mean time of performing maintenance on machine 2 is elapsed, the token c_{rn2} in $p_{combined}$ moves into p_2 by firing t_{r2}. When transition t_{r2} fires, it takes one token c_{rn2} from $p_{combined}$ and deposits it into p_2, denoting that a machine 2 recovery maintenance is finished. Finally, when an unreliable machine 3 breaks down in p_3, the token in p_3 moves into $p_{combined}$ by firing t_{f3}. If t_{f3} fires, it creates a token c_{rn3} and deposits it into place $p_{combined}$. If the mean time of performing maintenance on machine 3 is elapsed, the token c_{rn3} in $p_{combined}$ moves into p_3 by firing t_{r3}. When transition t_{r3} fires, it takes one token c_{rn3} from $p_{combined}$ and deposits it into p_3, denoting that a machine 3 recovery maintenance is finished.

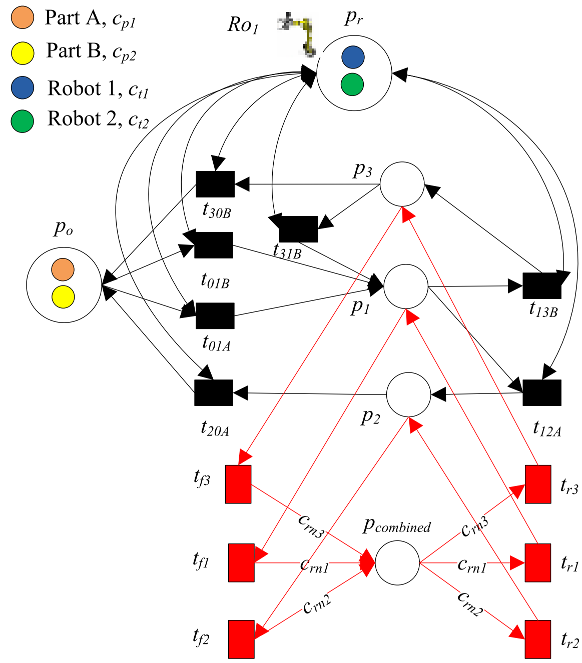

FIGURE 9. **Unreliable CROTPN for the model shown in Figure 6.**

TABLE 3. **Input variables for the model shown in Figure 10.**

Input	Description
x_1	The accelerometer
x_2	The current sensor
x_3	Strain gages
x_4	Coolant sensor
x_5	Acoustic emission sensor

D. ALGORITHM 4 APPLICATION

Finally, to illustrate the neural network and CROTPN for fault detection and treatment by using Algorithm 4, consider the unreliable CROTPN model presented in Figure 9. Algorithm 4 detects the types of various faults based on given parameters. The neural networks are used in two phases: training and testing. They learn how to identify the relation between inputs and outputs from the training phase. After training, the networks are tested using the test dataset. After the networks are tested and trained, they can diagnose faults under different operational conditions. The following concerns must be addressed when proposing models for system failure diagnosis: (a) data selection, (b) data normalization, and (c) structure network training and selection.

Using the CROTPN illustrated in Figure 9, we calculate the required data. The data consist of five input continuous factors [73]–[75] that are described in Table 3 and a single output is described in Table 4. The output variable from this

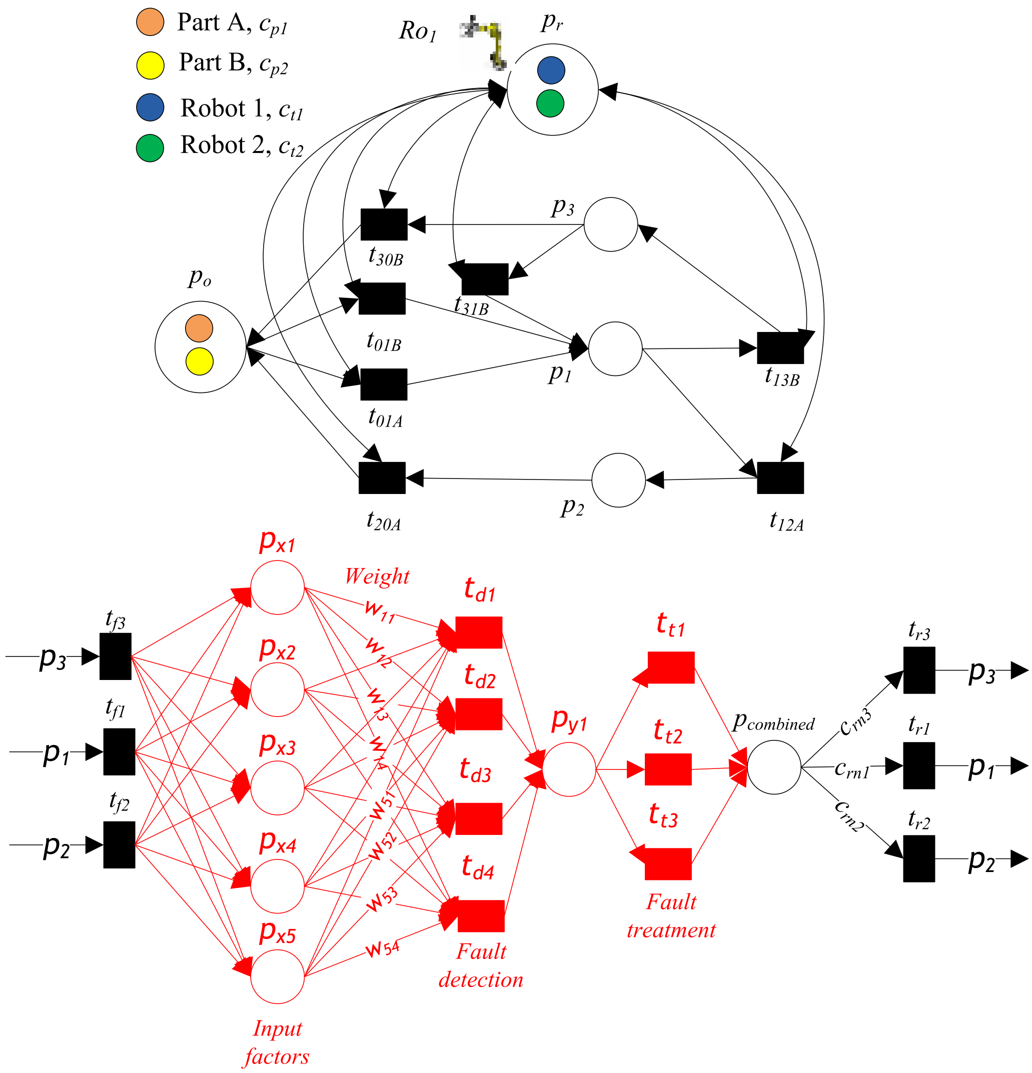

FIGURE 10. Neural unreliable CROTPN for the model shown in Figure 9.

TABLE 4. Output variables for the model shown in Figure 10.

Output	Description
y_1	Wearing the tool failure
y_2	Breaking the tool failure
y_3	Coolant failure
y_4	Programming errors

problem is obtained as [1 0 0 0] for fault type 1, [0 1 0 0] for fault type 2, [0 0 1 0] for fault type 3, and [0 0 0 1] for fault type 4. The GPenSIM code is employed to build the proposed neural unreliable CROTPN model. In Algorithm 4, a feature vector is obtained from the training dataset, the test dataset is used for fault diagnosis calculation, and the dataset for validation is used to interrupt iteration after a maximum capacity for generalization is reached. The collected datasets have 2250 patterns consisting of all five fault types. Moreover, the dataset includes 70% training patterns (1575 patterns), 20% test patterns (450 patterns), and 10% validation patterns (225 patterns). The training data differs totally from the testing data. Figure 10 displays the fault detection of an unreliable CROTPN model shown in Figure 9. Only a single unreliable system can fail at one time.

As shown in Figure 10, in the input layer, the extraction system consists of an accelerometer (x_1) that monitors mechanical vibrations, an electrical current sensor (x_2) that monitors variations in electrical motor current consumption, a strain gage (x_3) that monitors tool torsion or tool flexion, coolant sensor (x_4) that monitors the coolant level, and an acoustic emission sensor (x_5) that monitors acoustic effects of stress waves for tool break diagnosis. Acquisition systems collect signals from these sensors that identify machine tool states as being abnormal or normal. Random and uniform peaks are produced by the machine tool. Signals that exhibit peaks at the same wavelength signify slight tool-wear or erroneous programming of the machining parameters. The

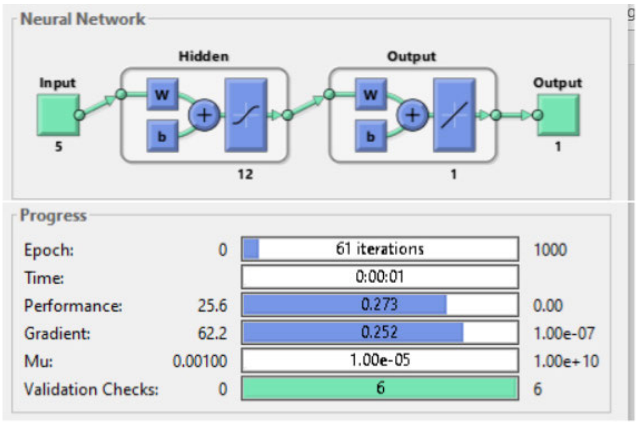

FIGURE 11. Performance of the neural unreliable CROTPN shown in Figure 10.

TABLE 5. Fault treatments for the model shown in Figure 10.

Fault treatment	Description
t_{t1}	Changing the parameter
t_{t2}	Changing the tool
t_{t3}	Changing the coolant

signals exhibiting random peaks indicate that the tool is very worn or broken [76].

As shown in Figure 10, the failures in the output layer are a result of tool wear (y_1), tool breaking (y_2), cooling failure (y_3), and programming mistakes (y_4). Failures produced from the machine or programming mistakes such as an inappropriate setting of parameters or overuse of tools result in this kind of uniform peaks. Random peaks may be produced from failures caused by tool-break. Failures that identified through uniform peaks and oscillations are caused by tool wear. Finally, coolant failures result from insufficient coolant.

Suggested treatments are presented in Table 5 to recover a failed machine. The treatments include changing the parameter (t_{t1}), changing the tool (t_{t2}), or changing the coolant (t_{t3}). If the output from the neural network is a tool failure, the parameter change is needed. If the output from the neural network is a tool break failure, the tool change is needed. When the output from the neural network is coolant failure, the coolant change is needed. In addition, if the output from the neural network is a machining parameter or programming error failure, the parameter change is needed. Several training experiments are performed to identify the best network structures and the parameters that will lead to minimal training errors. In addition, many training tests with varying numbers of hidden neurons are employed. The number of hidden neural network layers used is 12.

Figures 11 and 12 present the performance of the neural unreliable CROTPN model shown in Figure 10. The model can identify faults as a time function. The mean square error (MSE) generated by a model at 61 iterations with a learning rate of 0.00001 is 0.273 and leads to a 95% accurate model

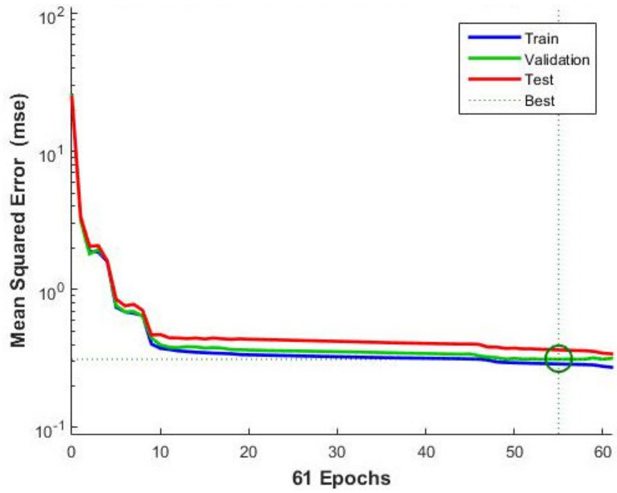

FIGURE 12. Learning performance of the neural unreliable CROTPN shown in Figure 10.

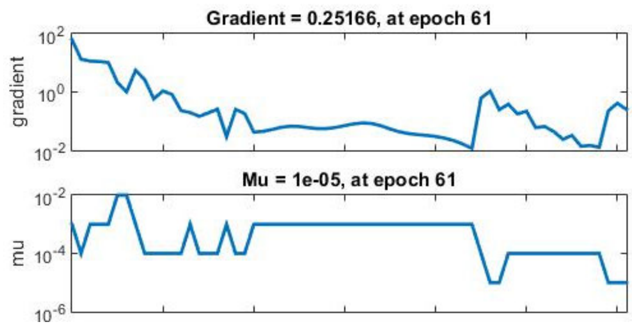

FIGURE 13. Validation check and learning rate of the neural unreliable CROTPN shown in Figure 10.

TABLE 6. Performance of the model shown in Figure 10.

Parameters	Value
Patterns of the training (%)	70
Patterns of the testing (%)	20
Patterns of the validation (%)	10
CPU time (s)	1
MSE	0.273
Value of learning rate	0.00001
Fault detection accuracy (%)	95

as shown in Table 6. Figure 13 indicates that the value of learning rate γ has a major impact on the convergence of the proposed model. Algorithm 4 converges slowly at low values of the learning rate, and it is very sensitive to the decreasing output if the learning rate is too high. The best solution for the coefficient of the correlation (R = 0.92031), as shown in Figure 14, indicates the efficiency of Algorithm 4. Furthermore, the value of the correlation coefficient properly represents the efficiency of fault diagnosis. Based on these findings, it is suggested that Algorithm 4 can be significantly integrated with regression when fault detection and treatment issues are addressed.

TABLE 7. Comparison of Algorithm 4 with the existing methods.

Parameter	Ref. [36]	Ref. [44]	Algorithm 3	Algorithm 4
Throughput	60	61	63	65
Throughput time	8.00	7.87	7.62	7.38
M1 utilization	52.92	55.21	54.38	54.17
M2 utilization	22.53	23.75	23.13	23.75
R1 utilization	46.68	48.33	47.08	48.75
R2 utilization	38.50	39.58	40	41.67

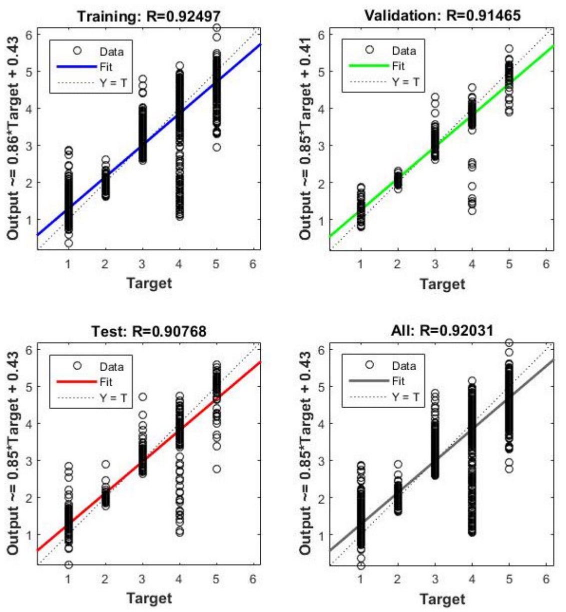

FIGURE 14. Regression results of the neural unreliable CROTPN model shown in Figure 10.

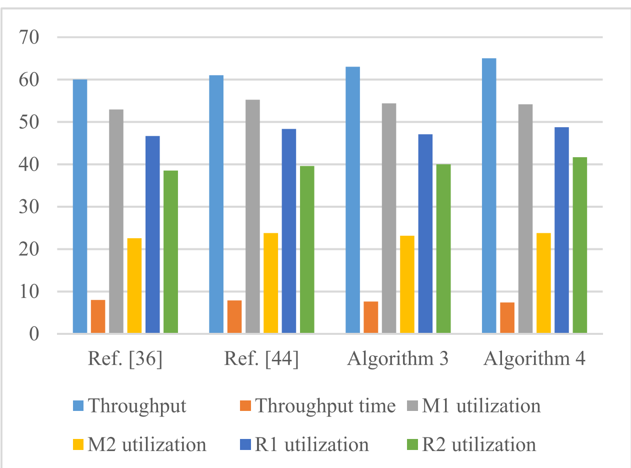

FIGURE 15. Time performance of Algorithm 4 compared with the current approaches for the model shown in Figure 10.

Finally, the GPenSIM code is employed to verify and validate Algorithm 4. The time performance criteria includes the total throughput time (min/part), total throughput (parts), and utilization of machines and robots (%). The experimental results are compared with those in [44], [36] and Algorithm 3. Consider the CROTPN shown in Figure 10. Table 7 illustrates the results with regard to the above time performance criteria. Algorithm 4 can achieve less throughput time, greater throughput, and better utilization than the other methods, as presented in Figure 15. Thus, Algorithm 4 is accurate.

VII. CONCLUSION

This paper proposes a four-step deadlock control policy for the diagnosis and treatment of faults for an RMS with unreliable resources. The first step involves the development of a CROTPN for rapid and effective reconfiguration of the RMS without considering resource failure. The second step presents sufficient and necessary conditions for ensuring the liveness of the CROTPN to guarantee that the model is live. The third step solves the failures of all resources in the CROTPN model and guarantees that the net is reliable by developing and adding a single recovery subnet to the CROTPN model at the second step. The fourth step designs a new hybrid method, which combines the CROTPN with neural networks for fault detection and treatment. The GPenSIM tool is used to assess the proposed strategy under the RMS configuration changes and the results are compared with existing methods in the literature.

The advantages of the proposed policy are as follows. (1) A CROTPN has a very compact structure, and the part production operations can be represented by adding colors compared with the studies in [77]–[79]. (2) The developed CROTPN can perform any complex RMS configuration compared with those in [77], [78], [80]. (3) It is more powerful and has a simpler structure compared with the research findings in [44]. (4) One common transportation resource place is designed to transport all part types in RMS compared with [80]. (5) It is a modular Petri net that can combine the CROTPN with neural networks. (6) It is suitable for resources of RMSs that are complex and sequential. (7) It implements a combined approach to ensure that no deadlock can occur; faults are detected, and treated in RMSs. (8) The proposed fault detection and treatment mode is verified and validated by a simulation study.

The main limitations of this research are that the developed CROTPN is based on discrete data types. Therefore, the future research of this paper will improve the developed method to design the CROTPN based on continuous data types to handle continuous RMS data. In addition, an automatic interface based on the developed algorithms has to be designed for dynamic changes in RMSs.

ACKNOWLEDGMENT

The authors are grateful to the Raytheon Chair for Systems Engineering for funding.

REFERENCES

[1] 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.

[2] H. Kaid, A. Al-Ahmari, and Z. Li, "Colored resource-oriented Petri net based ladder diagrams for PLC implementation in reconfigurable manufacturing systems," *IEEE Access*, vol. 8, pp. 217573–217591, 2020.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] M. Uzam, "An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions," *Int. J. Adv. Manuf. Technol.*, vol. 19, no. 3, pp. 192–208, Feb. 2002.

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

[10] Y.-L. Pan, C.-Y. Tseng, and T.-C. Row, "Design of improved optimal and suboptimal deadlock prevention for flexible manufacturing systems based on place invariant and reachability graph analysis methods," *J. Algorithms Comput. Technol.*, vol. 11, no. 3, pp. 261–270, Sep. 2017.

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

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

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

[14] 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.

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

[16] S. 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, May 2008.

[17] S. F. 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.

[18] 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.

[19] H. Yue, K. Xing, and Z. Hu, "Robust supervisory control policy for avoiding deadlock in automated manufacturing systems with unreliable resources," *Int. J. Prod. Res.*, vol. 52, no. 6, pp. 1573–1591, Mar. 2014.

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

[21] 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.

[22] 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.

[23] G. Liu, P. Li, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 7, pp. 1371–1385, Jul. 2019.

[24] N. Ran, J. Hao, S. Wang, Z. Dong, Z. He, Z. Liu, and Y. Ruan, "K-codiagnosability verification of labeled Petri nets," *IEEE Access*, vol. 7, pp. 185055–185062, 2019.

[25] E. A. Alzalab, Z. Yu, N. Wu, and H. Kaid, "Fault-recovery and repair modeling of discrete event systems using Petri nets," *IEEE Access*, vol. 8, pp. 170237–170247, 2020.

[26] 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. 137–141, Feb. 2003.

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

[28] 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.

[29] Y. Chen and Z. Li, "On structural minimality of optimal supervisors for flexible manufacturing systems," *Automatica*, vol. 48, no. 10, pp. 2647–2656, Oct. 2012.

[30] H. Kaid, A. Al-Ahmari, Z. Li, and R. Davidrajuh, "Automatic supervisory controller for deadlock control in reconfigurable manufacturing systems with dynamic changes," *Appl. Sci.*, vol. 10, no. 15, p. 5270, Jul. 2020.

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

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

[33] 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.

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

[35] H. Kaid, A. Al-Ahmari, and Z. Li, "Supervisor controller-based colored Petri nets for deadlock control and machine failures in automated manufacturing systems," *Int. J. Ind. Manuf. Eng.*, vol. 14, no. 10, pp. 426–432, 2020.

[36] 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.

[37] Y. Maki and K. A. Loparo, "A neural-network approach to fault detection and diagnosis in industrial processes," *IEEE Trans. Control Syst. Technol.*, vol. 5, no. 6, pp. 529–541, Nov. 1997.

[38] X.-Q. Liu, H.-Y. Zhang, J. Liu, and J. Yang, "Fault detection and diagnosis of permanent-magnet DC motor based on parameter estimation and neural network," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1021–1030, Oct. 2000.

[39] L. A. M. Riascos and P. E. Miyagi, "Supervisor system for detection and treatment of failures in manufacturing systems using distributed Petri nets," in *Proc. IFAC Workshop Manuf., Modelling, Manage. Control (MIM)*, Prague, Czech Republic, Aug. 2001, pp. 83–88.

[40] L. A. M. Riascos, F. G. Cozman, and P. E. Miyagi, "Detection and treatment of faults in automated machines based on Petri nets and Bayesian networks," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2003, pp. 729–734.
[41] 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.
[42] S. Rajakarunakaran, P. Venkumar, D. Devaraj, and K. S. P. Rao, "Artificial neural network approach for fault detection in rotary system," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 740–748, Jan. 2008.
[43] H. Honggui, L. Ying, and Q. Junfei, "A fuzzy neural network approach for online fault detection in waste water treatment process," *Comput. Electr. Eng.*, vol. 40, no. 7, pp. 2216–2226, Oct. 2014.
[44] 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.
[45] N. Wu and M. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 1, pp. 63–72, Feb. 2007.
[46] N. Wu and M. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1193–1202, Dec. 2005.
[47] N. Wu and M. Zhou, "Modeling and deadlock control of automated guided vehicle systems," *IEEE/ASME Trans. Mechatronics*, vol. 9, no. 1, pp. 50–57, Mar. 2004.
[48] H. Chen, N. Wu, and M. Zhou, "A novel method for deadlock prevention of AMS by using resource-oriented Petri nets," *Inf. Sci.*, vol. 363, pp. 178–189, Oct. 2016.
[49] N. Wu, M. Zhou, and Z. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 56–69, Jan. 2008.
[50] N. Wu and M. Zhou, *System Modeling and Control With Resource-Oriented Petri Nets*. New York, NY, USA: CRC Press, 2009.
[51] 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.
[52] N. Wu, "Avoiding deadlocks in automated manufacturing systems with shared material handling system," in *Proc. Int. Conf. Robot. Automat.*, 1997, pp. 2427–2432.
[53] N. Wu and M. Zhou, "Deadlock avoidance in semiconductor track systems," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 193–198.
[54] Z. Xiang, "Deadlock avoidance of flexible manufacturing systems by colored resource-oriented Petri nets with novel colored capacity," in *Proc. Int. Conf. Verification Eval. Comput. Commun. Syst.*, 2020, pp. 27–40.
[55] 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.
[56] N. Wu and M. Zhou, "Process vs resource-oriented Petri net modeling of automated manufacturing systems," *Asian J. Control*, vol. 12, no. 3, pp. 267–280, Apr. 2010.
[57] N. Wu, M. Zhou, and G. Hu, "On Petri net modeling of automated manufacturing systems," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, Apr. 2007, pp. 228–233.
[58] H. Chen, N. Wu, and M. Zhou, "Resource-oriented Petri net-based approach to deadlock prevention of AMSs," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 515–520.
[59] N. Wu and M. Zhou, "Resource-oriented Petri nets in deadlock avoidance of AGV systems," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2001, pp. 64–69.
[60] H. Chen, N. Wu, Z. Li, T. Qu, and H. Xiao, "Liveness of disjunctive and strict single-type automated manufacturing system: An ROPN approach," *IEEE Access*, vol. 7, pp. 17760–17771, 2019.
[61] 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.
[62] N. Wu, M. Zhou, and G. Hu, "One-step look-ahead maximally permissive deadlock control of AMS by using Petri nets," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–23, Jan. 2013.
[63] D. Y. Chao, "Fewer monitors and more efficient controllability for deadlock control in S^3PGR^2 (systems of simple sequential processes with general resource requirements)," *Comput. J.*, vol. 53, no. 10, pp. 1783–1798, Dec. 2010.
[64] X. Cong, C. Gu, M. Uzam, Y. Chen, A. M. Al-Ahmari, N. Wu, M. Zhou, and Z. Li, "Design of optimal Petri net supervisors for flexible manufacturing systems via weighted inhibitor arcs," *Asian J. Control*, vol. 20, no. 1, pp. 511–530, Jan. 2018.
[65] Q. Zhuang, W. Dai, S. Wang, J. Du, and Q. Tian, "An MIP-based deadlock prevention policy for siphon control," *IEEE Access*, vol. 7, pp. 153782–153790, 2019.
[66] 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.
[67] 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.
[68] 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.
[69] 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.
[70] Y. Hu, Z. Ma, and Z. Li, "Design of supervisors for active diagnosis in discrete event systems," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5159–5172, Dec. 2020.
[71] R. Davidrajuh, *Modeling Discrete-Event Systems With GPenSIM: An Introduction*. Cham, Switzerland: Springer, 2018.
[72] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.
[73] M. Santos, "Analysis of monitoring of tool-wear in frontal milling based on application of sensors," M.S. thesis, Escola Politécnica USP, São Paulo, Brazil, 1998, p. 184.
[74] V. Cunha, M. Santos, and C. Tu, "End mill sensor system for tool condition monitoring," Ph.D. dissertation, Escola Politécnica USP, São Paulo, Brazil, 2000, p. 136.
[75] G. Byrne, D. Dornfeld, I. Inasaki, G. Ketteler, W. König, and R. Teti, "Tool condition monitoring (TCM)—The status of research and industrial application," *CIRP Ann.*, vol. 44, no. 2, pp. 541–567, 1995.
[76] H. Asada and J.-J. Slotine, *Robot Analysis and Control*. Hoboken, NJ, USA: Wiley, 1986.
[77] S. Lee and D. M. Tilbury, "Deadlock-free resource allocation control for a reconfigurable manufacturing system with serial and parallel configuration," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 6, pp. 1373–1381, Nov. 2007.
[78] J. Li, X. Dai, and Z. Meng, "Automatic reconfiguration of Petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 156–167, Dec. 2009.
[79] Z. Yu, F. Guo, J. Ouyang, and L. Zhou, "Object-oriented Petri nets and π-calculus-based modeling and analysis of reconfigurable manufacturing systems," *Adv. Mech. Eng.*, vol. 8, no. 11, pp. 1–11, 2016.
[80] N. Wu and M. Zhou, "Intelligent token Petri nets for modelling and control of reconfigurable automated manufacturing systems with dynamical changes," *Trans. Inst. Meas. Control*, vol. 33, no. 1, pp. 9–29, Feb. 2011.

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

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

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

WADEA AMEEN received the M.Sc. and Ph.D. degrees from King Saud University, Saudi Arabia. He is currently an Assistant Professor with the Industrial Engineering Department, Faculty of Engineering and Architecture, Al-Yamamah University, Saudi Arabia. His research interests include additive manufacturing, CAD/CAM, product design analysis, and design of manufacturing systems, and optimization of manufacturing processes.