Technical Report

Design of Optimal Transition-based Supervisors for Flexible Manufacturing Systems via Resource Requirement Graphs

Yao Lu, Yufeng Chen, Zhiwu Li, and Christoforos N. Hadjicostis

Date: 2024-11-04

Design of Optimal Transition-based Supervisors for Flexible Manufacturing Systems via Resource Requirement Graphs

Yao Lu, Yufeng Chen, *Senior Member, IEEE*, Christoforos N. Hadjicostis, *Fellow, IEEE*, and Zhiwu Li, *Fellow, IEEE*

***Abstract*—In this article, we introduce a novel weighted digraph, namely a resource requirement graph, to address deadlock problems in flexible manufacturing systems (FMSs) modeled with Petri nets. A liveness-enforing supervisor consisting of recovery transitions designed to be enabled at pre-partial deadlocks is synthesized. First, a resource requirement graph is directly derived from a Petri net model of a flexible manufacturing system and is used to represent competition relations for shared resources by various processes. Subsequently, the notion of pre-partial deadlocks, described as a set of linear inequalities, is introduced by analyzing a resource requirement graph. Then, an algorithm is presented to develop a supervisor consisting of recovery transitions that are only enabled at the identified pre-partial deadlocks. The controlled Petri net model of an FMS under the designed supervisor is shown to be live with every initial reachable markings retained. The method proposed in this paper offers computational efficiency since the resource requirement graph is structurally compact compared to its counterpart net model. Finally, several examples are provided to demonstrate the developed techniques.**

***Index Terms*—Deadlock control, liveness, flexible manufacturing system, Petri net, resource requirement graph.**

I. Introduction

FLEXIBLE manufacturing systems (FMSs) [1], as typical automated processing systems, perform a crucial role in modern industrial production. An FMS is designed to concurrently perform various products in predetermined process sequences. The competition over shared but limited resources among various processes can lead to deadlocks [2], which are highly undesired situations where two or more processes interminably wait for others to release resources. Deadlocks in an FMS typically leads to unnecessary costs; thus, resolving deadlock problems in a highly automated FMS is an extensively researched problem.

Over the past decades, Petri nets are widely used to model, analyze, and manage FMSs. Typically, there are three deadlock resolution approaches: deadlock prevention [3], [4], [5], [26], [27], deadlock avoidance [6], [7], [8], [9], [28], [29], and deadlock detection and recovery [10], [11], [12], [30]. The first approach applies constraints on resource allocation for the purpose of averting the occurrence of deadlocks. The second approach is an on-line computation mechanism to handle deadlocks, where a proper processing stage of a job is chosen in each state to prevent a system from entering deadlocks. The third approach allows a deadlock to happen, however, once it appears and is detected, recovery strategies are implemented to bring a system into a live state.

Structural analysis [13], [14], [15], [16], [32], [35], [36] and reachability graph analysis [17], [18], [19], [33], [34] are two techniques of tackling deadlocks in Petri nets. Structural analysis develops deadlock prevention policies based on special structural items, such as siphons [14], [35] and resource-transition circuits (RTCs) [36], which are related to the liveness property of Petri nets. In this case, the obtained supervisor is suboptimal in general, since some legal markings have to be excluded from the controlled system in order to prevent deadlocks. Reachability graph analysis is considered as the most effective and detailed technique of handling deadlocks. The entire state evolution of a system can be reflected in a reachability graph. By locating deadlocks in a reachability graph, an optimal supervisor is often obtained without having to unnecessarily forbid legal markings. However, reachability graph analysis encounters the state explosion problem making it making it rather inefficient for complex systems.

Unlike the traditional deadlock controllers represented by places that stem from deadlock prevention policies, multiple studies have been done by proposing transition-based deadlock detection and recovery policies, where all original reachable markings are able to be preserved in the controlled systems. Even though these approaches are more behaviorally permissive compared to those based on places, almost all transition-based methods encounter the state explosion problem, since a reachability graph is required to be generated (at least once) for the purpose of detecting deadlocks. For example, the approach of computing recovery transitions in [20] relies on resolving the state equation to obtain the input places and output places of a recovery transition, but a complete reachability graph has to be generated. Moreover, this method tends to form a new livelock, in case it transforms a deadlock marking to a bad marking accidentally. Once a livelock happens, one has to step back and use the reachability graph to compute new valid

This work is supported in part by the Science and Technology Development Fund, MSAR, under Grants 0029/2022/AGJ and 0029/2023/RIA1, and in part by the Program of Guangdong under Grant 2023A0505020003. *(Corresponding author: Yufeng Chen.)*

Yao Lu is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: luyao@stu.xidian.edu.cn).

Yufeng Chen and Zhiwu Li are with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau, China. Zhiwu Li is also with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mails: chyf01@163.com; zhwli@xidian.edu.cn).

Christoforos N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus (e-mail: hadjicostis.christoforos@ucy.ac.cy).

transitions, until there is no livelock.

Chen *et al.* [21] propose two algorithms of designing recovery transitions. The first algorithm is capable of finding a small number of transitions by formulating an integer linear programming problem (ILPP) at each iterative step. The second algorithm is able to obtain the minimal number of transitions by formulating only one ILPP. Both methods are applicable to all FMS-oriented Petri net models. Dong *et al.* [22] manage to avoid solving ILPPs by proposing an iterative method utilizing a vector intersection approach. Although a live controlled net can be obtained without solving ILPPs, this approach still necessitates the enumeration of all deadlocks and some legal markings.

The method proposed in [23] manages to utilize structural properties of Petri nets and develop three iterative algorithms to avoid the need of generating a reachability graph. The experimental results show that the controlled system is live while preserving all reachable markings. However, such computation procedures become expensive since, in all the proposed three algorithms, the termination conditions must be verified with each iteration. Row and Pan [37] present a deadlock recovery policy, by which all reachable markings are capable of being accessed. In their related work in [38], a transition-based controller is synthesized utilizing the definition of crucial dead markings. Pan [24] develops an iterative approach for designing recovery transitions. The number of obtained transitions is fewer than in their prior work. However, reachability graphs must be recalculated at each iteration, as the fact that each iteration updates the model by adding a control transition. Therefore, the approach in [24] is not applicable for large-sized systems.

Our previous work [25] constructs a digraph, called a resource flow graph to design recovery transitions, in order to recover partial deadlocks. The controlled system is deadlock-free and retains all reachable states. More importantly, reachability graphs are not required and the method in [25] has polynomial time complexity. Nevertheless, its application is limited to specific ordinary Petri nets, e.g., S^{3}PR. In this paper, we aim to synthesize a liveness-enforcing supervisor composed of recovery transitions, which are only allowed to be enabled at pre-partial deadlocks; this can be achieved by constructing a weighted digraph called a resource requirement graph. Compared to [25], this method is applicable to more general Petri nets. By studying a resource requirement graph, the concept of pre-partial deadlocks is introduced (see Definition 7), which can be described as linear inequalities. Consequently, the controlled Petri net model of an FMS under the designed supervisor (composed by recovery transitions) is shown to be live and retains all original reachable markings.

The remainder of this paper is structured as follows. The overview of Petri nets is provided in Section II. Section III provides the definition of a resource requirement graph. Meanwhile, the procedure of finding pre-partial deadlocks by analyzing a resource requirement graph is also illustrated in this section. An algorithm for synthesizing a controller composed of recovery transitions that are allowed to be enabled at the pre-partial deadlocks is developed in Section IV, where it is proved that the net model, obtained by adding the designed recovery transitions, is live and retains all original reachable markings. Section V offers widely used benchmark examples to showcase the proposed approach. Finally, conclusions are drawn in Section VI.

We close this section by mentioning the conference paper [31], which this research extends. There are four main differences between [31] and this paper: first, a more exhaustive literature review is provided in this introductory section; second, a set of recovery transitions that are only enabled at pre-partial deadlocks is designed; third, this paper tackles both deadlock problems and livelock problems. The controlled net system is proved to be live and retains all original reachable markings; finally, the computational complexity of the proposed approach is analyzed.

II. Preliminary

A Petri net is a four-tuple $N = (P, T, F, W)$, where P is a finite (nonempty) set of places and T is a finite (nonempty) set of transitions with $P \cap T = \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is called the flow relation of a net, represented by arcs from places to transitions or from transitions to places; $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is a mapping that assigns a weight to an arc: $W(x, y) > 0$ if $(x, y) \in F$, otherwise $W(x, y) = 0$, where $x, y \in P \cup T$ and $\mathbb{N}$ denotes the set of non-negative integers. As shown in [3], [4], the places in a Petri net model of an FMS are usually classified into three types: $P = P^0 \dot{\cup} P_{\text{A}} \dot{\cup} P_{\text{R}}$, where P^0 is the set of idle places, P_{A} is the set of operation places, and P_{R} is the set of resource places. Therefore, a Petri net model of an FMS can also be denoted as $N = (P^0 \dot{\cup} P_A \dot{\cup} P_R, T, F, W)$.

A marking M of a Petri net N is a mapping from P to $\mathbb{N}$. The number of tokens in place p at making M is denoted as $M(p)$. Notation (N, M_0) is called a net system or a marked net and M_0 is called the initial marking of N.

The preset (postset) of a node $x \in P \cup T$ is defined as $^\bullet x = \{y \in P \cup T | (y, x) \in F\}$ ($x^\bullet = \{y \in P \cup T | (x, y) \in F\}$). Given a transition $t \in T$, a place in $^\bullet t$ is called an input place of t and a place in $t^\bullet$ is called an output place of t. Given a place $p \in P$, a transition in $^\bullet p$ is called an input transition of p and a transition in $p^\bullet$ is called an output transition of p.

A transition $t \in T$ is said to be enabled at a marking M if for all $p \in {}^\bullet t$, $M(p) \geq W(p, t)$, denoted as $M[t\rangle$. The firing of a transition t enabled at M yields a marking M' such that for all $p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$, denoted by $M[t\rangle M'$. The set of markings reachable from M in N is called the reachability set of Petri net system (N, M), denoted as $R(N, M)$. Given a net system (N, M_0), $t \in T$ is live at M_0 if for all $M \in R(N, M_0)$, there exists $M' \in R(N, M)$ such that $M'[t\rangle$ holds. Net system (N, M_0) is live if for all $t \in T$, t is live at M_0. A marking M is called a deadlock if there does not exist $t \in T$ such that $M[t\rangle$ holds. Net system (N, M_0) is deadlock-free (weakly live or livelock) if for all $M \in R(N, M_0)$, there exists $t \in T$ such that $M[t\rangle$ holds. A marking M is said to be a partial deadlock if there exists a nonempty subset $T' \subseteq T$ satisfying for all $M' \in R(N, M)$, there does not exist $t \in T'$ such that $M'[t\rangle$ holds.

The output and input incidence matrices of a Petri net are defined as $[N^+](p, t) = W(t, p)$ and $[N^-](p, t) = W(p, t)$, for all $p \in P$ and $t \in T$, respectively. The incidence matrix of a net is defined as $[N] = [N^+] - [N^-]$, where $[N](p,t) = W(t,p) - W(p,t)$. Given a place p (a transition t), its incidence vector, a row (column) in $[N]$ is denoted by $[N](p,\cdot)$ ($[N](\cdot,t)$).

A P-vector is a column vector $I : P \to \mathbb{Z}$ indexed by P, where $\mathbb{Z}$ is the set of integers. P-vector I is called a place invariant (P-invariant) if $I \neq \mathbf{0}$ and $I^T[N] = \mathbf{0}^T$. P-invariant I is called a P-semiflow if every element of I is non-negative. $||I|| = \{p | I(p) \neq 0\}$ is called the support of I. A P-invariant I is said to be a minimal P-invariant if $||I||$ is not a superset of the support of any other P-invariant and its components are mutually prime.

III. Deadlock Detection Using Resource Requirement Graphs

In this section, we first introduce the definition of a resource requirement graph, which is derived from a Petri net model of an FMS. A small example is also provided to demonstrate the process of constructing such a graph. Then, we argue that the pre-partial deadlocks can be described as a set of linear inequalities by analyzing a resource requirement graph.

Definition 1: Let P_A^* be the set of finite-length sequences of operation places in a Petri net $N = (P^0 \dot\cup P_A \dot\cup P_R, T, F, W)$. A sequence $\eta = y_1 y_2 \ldots y_n \in P_A^*$ is called a production sequence of N, if the following conditions hold:

- for all $i \in \{1, 2, \ldots, n-1\}$, there exists a transition $t_i \in {y_i}^\bullet \cap {}^\bullet y_{i+1}$,
- there exists an idle place $p_0 \in {}^{\bullet\bullet}y_1 \cap {y_n}^{\bullet\bullet}$. □

Compared with our previous work [25], the deadlock detection and recovery strategy to be proposed in this article has a wider application scope, which targets generalized Petri nets that satisfy the conditions below.

1) Any idle place p_0 is associated with a minimal P-semiflow I_{p_0} such that for all places $p \in ||I_{p_0}|| \backslash \{p_0\}$, $p \in P_A$ holds.
2) Any resource place r is associated with a minimal P-semiflow I_r such that for all places $p \in ||I_r|| \setminus \{r\}$, $p \in P_A$ holds.
3) For all $t \in T$, $|{}^\bullet t \cap P_A| \leq 1$.
4) Let y be an operation place. If $|y^\bullet| \geq 2$, for any two transitions t_k, $t_l \in y^\bullet$, $k \neq l$, $|{}^\bullet t_k \cap P_R| \leq 1$ and $|{}^\bullet t_l \cap P_R| \leq 1$.

Condition 1) (Condition 2)) indicates that for each idle place (resource place) in a Petri net, there is an associated P-semiflow such that its support only consists of some operation places and the idle place (resource place) itself. In other words, there are no other idle places and resource places involved in the P-semiflow. Conditions 1) and 2) are satisfied by all manufacturing-oriented Petri net models proposed in the literature as far as the authors know. Condition 3) implies that for each transition, its preset should contain one operation place at most. Condition 4) indicates that if there is more than one transition in the postset of an operation place, then for any transition that is in the postset of the operation place, the preset of the transition should contain only one resource place at most. For the sake of intuition, conditions 3) and 4) can be visually represented by Figs. 1 and 2, respectively. Both conditions 3) and 4) are satisfied by removing the blue colored places and arcs, where y_i (resp., x_i and t_i) is an operation place (resp., a resource place and a transition), $i \in \{1, 2, \ldots, n\}$.

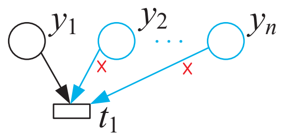


Fig. 1: Illustration of condition 3).

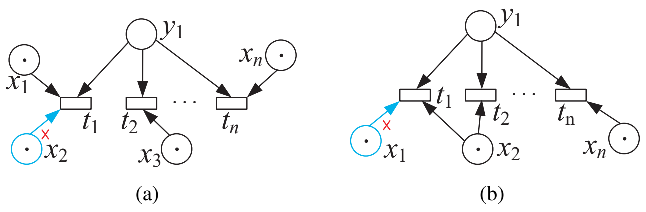


Fig. 2: Illustration of condition 4).

A. Resource Requirement Graph

For a Petri net model of an FMS, the competition over shared resources among various processes can lead to deadlocks. In order to identify deadlocks without requiring a reachability graph, we propose a straightforward way to obtain the conflict among different processes by deriving a digraph, namely a resource requirement graph, from a Petri net. A resource requirement graph can directly depict the amount of resource units and types required by each process.

Definition 2: Given a Petri net $N = (P^0 \dot\cup P_A \dot\cup P_R, T, F, W)$, a resource requirement graph $G = (P_G, \chi, E, \gamma, S, \phi)$ associated with N is a digraph such that:

- $P_G = P_A \cup P_R$.
- $\chi : P_G \to \mathbb{N}$ is a mapping from P_G to $\mathbb{N}$: $\chi(p) = M(p)$, where $p \in P_G$ and M is a marking of N.
- $E \subseteq (P_R \times P_A) \cup (P_A \times P_R)$ is called its resource flow relation depicted by directed arcs from resource places to operation places or vice versa. For all $x \in P_R$ and $y \in P_A$,
  $(x, y) \in E$, if there exists a transition $t \in x^\bullet \cap {}^\bullet y$,
  $(y, x) \in E$, if there exists a transition $t \in y^\bullet \cap x^\bullet$.
- $\gamma : E \to \mathbb{N}$ is a mapping that assigns a weight to an arc in E.
  $\gamma(x, y) = \min\{W(x,t) | t \in x^\bullet \cap {}^\bullet y\}$, if $(x, y) \in E \cap (P_R \times P_A)$, otherwise, $\gamma(x, y) = 0$,
  $\gamma(y, x) = \min\{W(x,t) | t \in x^\bullet \cap y^\bullet\}$, if $(y, x) \in E \cap (P_A \times P_R)$, otherwise, $\gamma(y, x) = 0$.
- $S \subseteq P_A \times P_A$ is a set of operation place pairs. Such a pair is graphically represented by a dashed arc with an arrow from one operation place to another. For a production sequence $\eta = y_1 y_2 \cdots y_n$ of N, for all $i \in \{1, 2, \ldots, n-1\}$, we have $(y_i, y_{i+1}) \in S$.

- $\phi : S \to 2^{(P_R \times \mathbb{N})}$ is a mapping that assigns a set of ordered pairs of resource places and weights to an arc in S. Given a production sequence $\eta = y_1 y_2 \cdots y_n$ of N, for all $i \in \{1, 2, \ldots, n-1\}$, it holds $(y_i, y_{i+1}) \in S$. Mapping ϕ can be recursively defined as follows:
  1) for $i = 1$, $\phi((y_1, y_2)) = \{(x, z_x^{y_2}) | \exists x \in P_R : W(x, t_0) > 0 \wedge z_x^{y_2} > 0\}$, where $z_x^{y_2} = \min\{W(x, t_0)\} + W(x, t_1) - W(t_1, x)$ and $t_0 \in {}^\bullet y_1 \cap T$. Arc (x, y_2) is removed from E if $z_x^{y_2} > 0$,
  2) for all $i \in \{2, 3, \ldots, n-1\}$, $\phi((y_i, y_{i+1})) = \{(x, z_x^{y_{i+1}}) | \exists x \in P_R : W(x, t_{i-1}) > 0 \wedge z_x^{y_i} = 0 \wedge z_x^{y_{i+1}} > 0\} \cup \{(x', z_{x'}^{y_{i+1}}) | \exists x' \in P_R : z_{x'}^{y_i} > 0 \wedge z_{x'}^{y_{i+1}} > 0\}$, where $z_x^{y_{i+1}} = W(x, t_{i-1}) + W(x, t_i) - W(t_i, x)$, $z_{x'}^{y_{i+1}} = z_{x'}^{y_i} + W(x', t_i) - W(t_i, x')$, and $t_i \in y_i^\bullet \cap {}^\bullet y_{i+1}$. Arc (x, y_{i+1}) $((x', y_{i+1}))$ is removed from E if $z_x^{y_{i+1}} > 0$ $(z_{x'}^{y_{i+1}} > 0)$.

Given a Petri net, its resource requirement graph can be constructed following Definition 2. Note that there are only resource places and operation places in a resource requirement graph. Therefore, the number of the nodes of a resource requirement graph G is $|P_R| + |P_A|$, which is $|P^0| + |T|$ less than the number of the nodes of its counterpart net model N. Moreover, a resource requirement graph is capable of illustrating the competition over limited resources among various processes by revealing the direct relation between resource places and operation places. Hence, the number of edges of a resource requirement graph G is at most half of that of N.

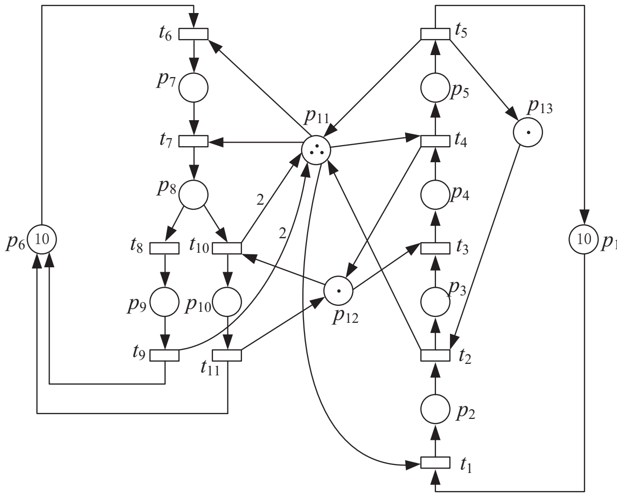


Fig. 3: Petri net model N_1.

Next, we provide an example to demonstrate how to derive a resource requirement graph from a Petri net. Let us consider Petri net N_1 in Fig. 3, where $P^0 = \{p_1, p_6\}$, $P_R = \{p_{11}, p_{12}, p_{13}\}$, and $P_A = \{p_2 - p_5, p_7 - p_{10}\}$. There are three production sequences in N_1: $\eta_1 = p_2 p_3 p_4 p_5$, $\eta_2 = p_7 p_8 p_9$, and $\eta_3 = p_7 p_8 p_{10}$. Its resource requirement graph G_1 is generated following Definition 2, as depicted in Fig. 4. A few specific processes are presented as follows.

P_G: $P_G = P_A \cup P_R = \{p_2 - p_5, p_7 - p_{13}\}$.

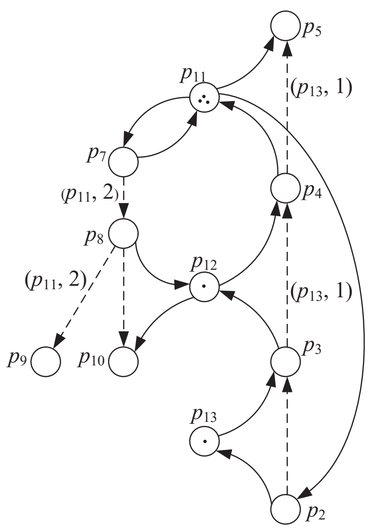


Fig. 4: Resource Requirement Graph G_1.

χ: The initial marking of N_1 is $M = [10, 0, 0, 0, 0, 10, 0, 0, 0, 0, 3, 1, 1]^T$. For instance, we have $\chi(p_2) = M(p_2) = 0$ and $\chi(p_{11}) = M(p_{11}) = 3$.

E: By $t_6 \in p_{11}{}^\bullet \cap {}^\bullet p_7$, we have $(p_{11}, p_7) \in E$. Analogously, due to $t_7 \in p_7{}^\bullet \cap p_{11}{}^\bullet$, we have $(p_7, p_{11}) \in E$. The remainder of E can be generated accordingly by using the same method.

γ: For arc $(p_{11}, p_7) \in E$, we have $\gamma(p_{11}, p_7) = \min\{W(p_{11}, t) | t \in p_{11}{}^\bullet \cap {}^\bullet p_7\} = \min\{W(p_{11}, t) | t \in \{t_6\}\} = W(p_{11}, t_6) = 1$. For arc $(p_7, p_{11}) \in E$, we have $\gamma(p_7, p_{11}) = \min\{W(p_{11}, t) | t \in p_7{}^\bullet \cap p_{11}{}^\bullet\} = \min\{W(p_{11}, t) | t \in \{t_7\}\} = W(p_{11}, t_7) = 1$. The rest are not detailed due to space considerations.

S: We have $(p_2, p_3) \in S$, $(p_3, p_4) \in S$, and $(p_4, p_5) \in S$ by $\eta_1 = p_2 p_3 p_4 p_5$. Furthermore, we have $(p_7, p_8) \in S$ and $(p_8, p_9) \in S$ $((p_8, p_{10}) \in S)$ by $\eta_2 = p_7 p_8 p_9$ $(\eta_3 = p_7 p_8 p_{10})$.

ϕ: Let us consider $\eta_2 = p_7 p_8 p_9 = y_1 y_2 y_3$. We have $t_0 \in {}^\bullet y_1 \cap T = {}^\bullet p_7 \cap T = \{t_6\}$, $t_1 \in y_1{}^\bullet \cap {}^\bullet y_2 = p_7{}^\bullet \cap {}^\bullet p_8 = \{t_7\}$, and $t_2 \in y_2{}^\bullet \cap {}^\bullet y_3 = p_8{}^\bullet \cap {}^\bullet p_9 = \{t_8\}$. By its definition, $\phi((p_7, p_8))$ is defined as a set $\{(x, z_x^{p_8}) | \exists x \in P_R : W(x, t_6) > 0 \wedge z_x^{p_8} > 0\}$

It is noted that there exists only one resource place p_{11} such that $W(p_{11}, t_6) > 0$. As a result, we have $z_{p_{11}}^{p_8} = z_x^{y_2} = \min\{W(p_{11}, t_6)\} + W(p_{11}, t_7) - W(t_7, p_{11}) = 1 + 1 - 0 = 2 > 0$. Therefore, we have $\phi((p_7, p_8)) = \{(p_{11}, 2)\}$. Arc (p_{11}, p_8) is removed from E due to $z_{p_{11}}^{p_8} = 2 > 0$.

Similarly, $\phi((p_8, p_9))$ is defined as a set $\{(x, z_x^{p_9}) | \exists x \in P_R : W(x, t_7) > 0 \wedge z_x^{p_8} = 0 \wedge z_x^{p_9} > 0\} \cup \{(x', z_{x'}^{p_9}) | \exists x' \in P_R : z_{x'}^{p_8} > 0 \wedge z_{x'}^{p_9} > 0\}$. It is noted that there is not a resource place such that $W(x, t_7) > 0$ and $z_x^{p_8} = 0$ hold. However, we know that $z_{p_{11}}^{p_8} > 0$ and $z_{p_{11}}^{p_9} = z_{x'}^{y_3} = z_{p_{11}}^{p_8} + W(p_{11}, t_8) - W(t_8, p_{11}) = 2 + 0 - 0 = 2 > 0$. Hence, we have $\phi((p_8, p_9)) = \emptyset \cup \{(p_{11}, 2)\} = \{(p_{11}, 2)\}$ The rest of the resource requirement graph can be constructed according to Definition 2.

Given a resource requirement graph, according to Definition 2, operation place y, representing a processing stage, requires resources from resource place x if there is an arc from x to y or $z_x^y > 0$ in the resource requirement graph. The quantity of resource units required by y from x is $\gamma(x,y)$ or z_x^y. Moreover, in order to proceed to the next process of y, operation place y requires resources from resource place x if there is an arc from y to x in the resource requirement graph. The amount of resource units that y requires from x is $\gamma(y,x)$. For example, operation place p_7 in Fig. 4 requires one resource unit from resource place p_{11}, as there is an arc from p_{11} to p_7 and $\gamma(p_{11},p_7)=1$. To proceed to the next process of p_7, i.e., p_8, operation place p_7 also needs one resource unit from p_{11} since there is an arc from p_7 to p_{11} and $\gamma(p_7,p_{11})=1$.

Definition 3: Given a Petri net $N=(P^0\dot{\cup}P_A\dot{\cup}P_R,T,F,W)$, an operation place y is said to be a release-free operation place if there exists a transition $t\in y^\bullet$ such that ${}^\bullet t\cap P_R=\emptyset$. The release-free set of N, denoted as $\mathcal{R}_\text{f}$, is defined as the set of all release-free operation places in N.

In Definition 3, each release-free operation place has at least one output transition such that the preset of the output transition does not include any resource place. According to the firing rules of Petri nets, the output transition can be enabled without requiring any resource. In this case, the tokens currently held by a release-free operation place can be unconditionally released. Let us consider the Petri net in Fig. 3. According to Definition 3, the release-free set of N_1 is $\mathcal{R}_\text{f}=\{p_5,p_8,p_9,p_{10}\}$. For example, operation place p_5 is a release-free operation place since there exists a transition $t_5\in {p_5}^\bullet$ such that ${}^\bullet t_5\cap P_R=\emptyset$. In words, once p_5 holds a token, the token can be released without requiring any resource.

Definition 4: Given a Petri net $N=(P^0\dot{\cup}P_A\dot{\cup}P_R,T,F,W)$ and its resource requirement graph $G=(P_G,\chi,E,\gamma,S,\phi)$, the alternative release set of an arc $(y,x)\in E\cap(P_A\times P_R)$ is defined as $\Delta=\{(y,x')\in E|y\notin\mathcal{R}_\text{f},\nexists t\in y^\bullet\cap x^\bullet: t\in y^\bullet\cap x'^\bullet\}$, where $\mathcal{R}_\text{f}$ is the release-free set of N.

On the other hand, in case of preventing an operation place from releasing its tokens, all output transitions of the operation place should be disabled. For an arc $(y,x)\in E\cap(P_A\times P_R)$, in order to prevent the tokens held by y from being released, we should disable transition $t\in y^\bullet\cap x^\bullet$ and transition $t'\in y^\bullet\cap x'^\bullet$, for all $(y,x')\in\Delta$. Let us consider the Petri net in Fig. 6(a). Its resource requirement graph is depicted in Fig. 6(b). According to Definition 4, the alternative release set of arc (p_1,p_7) is $\delta_1=\{(p_1,p_8)\}$. There are two output transitions of p_1, namely transitions t_2 and t_3. To prevent the tokens held by p_1 from being released, both $t_2\in {p_1}^\bullet\cap {p_7}^\bullet$ and $t_3\in {p_1}^\bullet\cap {p_8}^\bullet$ should be disabled.

B. Resource-Operation Loop

Definition 5: Given a resource requirement graph $G=(P_G,\chi,E,\gamma,S,\phi)$, an operation pattern of G is defined as $\delta=(b_1,b_2,\dots,b_n)$, $n\in\{1,2,\dots\}$, such that:

- $\delta=(b_1)$ and b_1 is an operation place, if $n=1$,
- for all $i\in\{1,2,\dots,n-1\}$, $(b_i,b_{i+1})\in S$ and there does not exist $x\in P_R$ such that $(b_i,x)\in E$ and $(x,b_{i+1})\in E$, if $n\geq 2$.

Definition 6: Given a resource requirement graph $G=(P_G,\chi,E,\gamma,S,\phi)$, a resource-operation loop of G is defined as $\beta=a_1\rightarrow\delta_1\rightarrow a_2\rightarrow\delta_2\rightarrow\cdots\rightarrow a_m\rightarrow\delta_m\rightarrow a_1$ with operation pattern $\delta_k=(b_{k1},b_{k2},\dots,b_{kn_k})$ $(k\in\{1,2,\dots,m\})$, such that:

- for all $i\in\{1,2,\dots,m\}$, $(a_i,b_{i1})\in E$,
- for all $i\in\{1,2,\dots,m-1\}$, $(b_{in_i},a_{i+1})\in E$,
- $(b_{mn_m},a_1)\in E$,

where $a_i\in P_R$ and n_k and m are positive integers.

By Definitions 5 and 6, there are three resource-operation loops in Fig. 4, namely, $\beta_1=p_{11}\rightarrow(p_7)\rightarrow p_{11}$, $\beta_2=p_{11}\rightarrow(p_7,p_8)\rightarrow p_{12}\rightarrow(p_4)\rightarrow p_{11}$, and $\beta_3=p_{11}\rightarrow(p_2)\rightarrow p_{13}\rightarrow(p_3)\rightarrow p_{12}\rightarrow(p_4)\rightarrow p_{11}$. Let E_{rd} be the subset of E obtained by removing the arcs from E that do not appear in any resource-operation loop. In what follows, when we talk about a resource-operation loop $\beta=a_1\rightarrow\delta_1\rightarrow a_2\rightarrow\delta_2\rightarrow\cdots\rightarrow a_m\rightarrow\delta_m\rightarrow a_1$, we use $a_i\in\beta$, $b_{ij}\in\beta$, and $\delta_i\in\beta$ to denote, respectively, that a resource place, an operation place, and an operation pattern are contained in β.

C. Pre-partial deadlock

It is shown in this section that the competition over shared resources among various processes in an FMS can be represented by resource-operation loops. Meanwhile, we concentrate on resource-operation loops to identify all potential pre-partial deadlocks, which can be described as a set of linear inequalities. An algorithm is presented to distinguish actual pre-partial deadlocks from potential ones. First, the notion of a pre-partial deadlock marking is defined as follows.

Definition 7: Given a Petri net $N=(P,T,F,W)$ and the set of all partial deadlock markings $\mathcal{M}_\text{PD}$, a marking M is called a pre-partial deadlock marking if it satisfies:

1) there exists a nonempty subset $T'\subseteq T$ such that for all $M'\in R(N,M)$, there does not exist a transition $t\in T'$ such that $M'[t\rangle$, and
2) there exists a sequence of transitions $\sigma\in T^*$ such that $M[\sigma\rangle M_{pd}$, where $M_{pd}\in\mathcal{M}_\text{PD}$.

Definition 7 states that a pre-partial deadlock marking M is capable of reaching a partial deadlock by firing the sequence of transitions σ at M. In words, a pre-partial deadlock marking can inevitably lead to a partial deadlock in finite steps. Furthermore, the sequence σ can be an empty sequence, which signifies that the pre-partial deadlock marking M is already a partial deadlock. Thus, based on the definition of partial deadlocks, the set of partial deadlocks $\mathcal{M}_\text{PD}$ is a subset of the set of pre-partial deadlocks $\mathcal{M}_\text{PPD}$, i.e., $\mathcal{M}_\text{PD}\subseteq\mathcal{M}_\text{PPD}$. In this case, all partial deadlocks can be recovered as long as we recover all pre-partial deadlocks.

By focusing on resource-operation loops of a resource requirement graph, we intend to find a linearly ordered processes where each process requests resources currently processes by the next process. Meanwhile, pre-partial deadlocks can be found by purposely assigning system resources of each resource-operation loop such that no process in a linearly ordered process is capable of releasing the resources held by itself until it gets the resources held by the next process.

Let $\beta=a_1\rightarrow\delta_1\rightarrow a_2\rightarrow\delta_2\rightarrow\cdots\rightarrow a_m\rightarrow\delta_m\rightarrow a_1$ be a resource-operation loop with operation pattern $\delta_k=(b_{k1},b_{k2},\dots,b_{kn_k})$ $(k\in\{1,2,\dots,m\})$, where n_k and

m are positive integers. Any reachable marking, including a pre-partial deadlock M, must satisfy the property of resource conservativeness during resource allocation. That is to say, for a resource place $a_i \in \beta$, after allocating its resource units to the operation places that require resources from a_i, the weighted sum of tokens distributed among all the operation places must be less than or equal to the initial marking of the resource place a_i, i.e.,

$$\sum_{l\in\mathbb{W}}\gamma(a_i,y_l)\cdot M(y_l)+\sum_{j\in\mathbb{V}}z_{a_i}^{y_j}\cdot M(y_j)\leq M_0(a_i),\quad \forall i\in\{1,2,\dots,m\},\tag{1}$$

where $\mathbb{W}=\{l|(a_i,y_l)\in E_{rd}\}$ and $\mathbb{V}=\{j|z_{a_i}^{y_j}>0\}$ are two index sets, and $M_0(a_i)$ is the initial marking of a_i. According to Definition 2, the amount of resource units that operation place y_l (y_j) (operation places represent processing stages) needs from a_i is denoted by $\gamma(a_i,y_l)$ ($z_{a_i}^{y_j}$). For instance, operation place p_7 in Fig. 4 needs $\gamma(p_{11},p_7)=1$ resource unit from p_{11}, while operation place p_8 in Fig. 4 needs $z_{p_{11}}^{p_8}=2$ resource units from p_{11}. Moreover, operation place p_7 requires $\gamma(p_7,p_{11})=1$ resource unit from p_{11} to release the resource p_7 currently holds. Operation place p_8 needs $\gamma(p_8,p_{12})=1$ resource unit from p_{12} to release the resource p_8 currently holds.

Eq. (1) provides all the possible values of the markings of y_l and y_j, i.e., $M(y_j)$ and $M(y_l)$, by only considering resource place a_i. However, an operation place can simultaneously need multiple resource units from various resource types in a Petri net. For the purpose of precisely determining the possible values of the markings, the restriction over the markings of y_l and y_j should also consider other resource places that they require, i.e.,

$$M(y_l)\leq\min\{\lfloor\frac{1}{\gamma(a_s,y_l)}\cdot M_0(a_s)\rfloor,\lfloor\frac{1}{z_{a_t}^{y_l}}\cdot M_0(a_t)\rfloor \;|(a_s,y_l)\in E, a_s\neq a_i, z_{a_t}^{y_l}>0\}\quad \forall l\in\mathbb{W},\forall i\in\{1,2,\dots,m\},\tag{2}$$

$$M(y_j)\leq\min\{\lfloor\frac{1}{\gamma(a_s',y_j)}\cdot M_0(a_s')\rfloor,\lfloor\frac{1}{z_{a_t'}^{y_j}}\cdot M_0(a_t')\rfloor \;|(a_s',y_j)\in E, z_{a_t'}^{y_j}>0, a_t\neq a_i\}\quad \forall j\in\mathbb{V},\forall i\in\{1,2,\dots,m\}.\tag{3}$$

Eqs. (2) and (3) give the maximum markings that y_l and y_j can reach by considering other resources they need. For example, operation place p_4 in Fig. 4 needs one resource unit from p_{12} and one resource unit from p_{13}. Hence, we have $M(p_4)\leq\min\{\lfloor\frac{1}{\gamma(p_{12},p_4)}M_0(p_{12})\rfloor,\lfloor\frac{1}{z_{p_{13}}^{p_4}}M_0(p_{13})\rfloor\}=\min\{1,1\}=1$. Now, let us consider this situation where we modify the initial marking of p_{13} from 1 to 2. Suppose that we already apply Eq. (1) to each resource place in β_3. For p_{13}, we have $M(p_3)+M(p_4)\leq 2$, from which we can see that the maximum marking of p_4 can be 2. However, p_4 also requires resources from p_{12}. By Eq. (3), we have $M(p_4)\leq\min\{\lfloor\frac{1}{\gamma(p_{12},p_4)}M_0(p_{12})\rfloor\}=1$. In this case, the maximum marking of p_4 should be less than or equal to 1 instead of 2, which follows from the property of resource conservativeness. Nonetheless, this analysis does not imply that the maximum marking is reachable with the evolution of the system, since Eqs. (2) and (3) hold only due to resource conservativeness. In real cases, it is much more complicated as the system's evolution must follow the firing rules of Petri nets. In other words, there may exist some spurious markings being calculated.

Let us clarify how spurious markings may be calculated by considering the small Petri net depicted in Fig. 5(a), where $P_R=\{p_3\}$ and $P_A=\{p_1,p_2\}$. According to Definition 2, a resource requirement of the Petri net can be constructed, as depicted in Fig. 5(b). By Eq. (1), we have $2M(p_1)+M(p_2)\leq 2$. Clearly, there are four feasible solutions that satisfy the inequalities, which are $0p_1+0p_2$, p_1+0p_2, $0p_1+p_2$, and $0p_1+2p_2$. The first three markings can be achieved by firing empty string ε, t_1, and t_1t_2, respectively. However, it is impossible to reach $0p_1+2p_2$, since t_2 has to be fired twice. After firing t_1t_2, we have $M(p_1)=0$, $M(p_2)=1$, and $M(p_3)=1$. In this case, both t_1 and t_2 cannot be enabled anymore. Although our method may compute certain spurious markings that are not reachable in the actual system, it is acceptable for the purpose of deadlock recovery, as the spurious markings are treated the same as the real pre-partial deadlocks markings. A disadvantage of this approach, however, is that it might obtain redundant recovery transitions in order to address pre-partial deadlocks at these spurious markings.

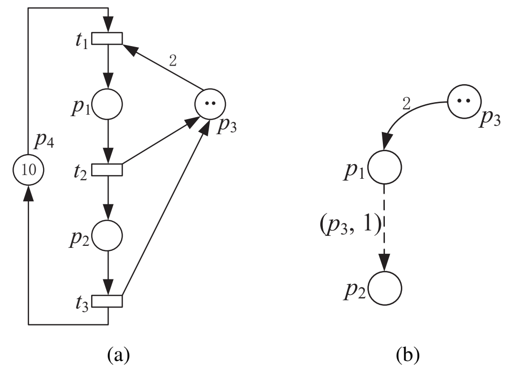


Fig. 5: (a) Petri net model, (b) resource requirement graph.

For a pre-partial deadlock M, there must exist a set of operation places that are incapable of releasing the resource held by themselves, which indicates that the amount of tokens left in a_i (a_1), $i\in\{2,3,\dots,m\}$, after resource allocation, is not big enough to release the resources that are held by $b_{i-1n_{i-1}}$ (b_{mn_m}). Therefore, we have

$$M_0(a_i)-\sum_{l\in\mathbb{W}}\gamma(a_i,y_l)\cdot M(y_l)-\sum_{j\in\mathbb{V}}z_{ai}^{y_j}\cdot M(y_j)\leq\gamma(b_{mn_m},a_i)-1, i=1,\tag{4}$$

$$M_0(a_i)-\sum_{l\in\mathbb{W}}\gamma(a_i,y_l)\cdot M(y_l)-\sum_{j\in\mathbb{V}}z_{ai}^{y_j}\cdot M(y_j)\leq\gamma(b_{i-1n_{i-1}},a_i)-1,\forall i\in\{2,3,\dots,m\}.\tag{5}$$

Eq. (4) (Eq. (5)) is active only if $M(b_{mn_m})>0$ ($M(b_{i-1n_{i-1}})>0$). To this end, a binary variable d_i ($i\in$

$\{1, 2, \ldots, m\}$) is introduced for b_{in_i}. Eqs. (4) and (5) can be transformed into

$$M_0(a_i) - \sum_{l \in \mathbb{W}} \gamma(a_i, y_l) \cdot M(y_l) - \sum_{j \in \mathbb{V}} z_{ai}^{y_j} \cdot M(y_j) \leq \gamma(b_{mn_m}, a_i) - 1 + Q \cdot (1 - d_m), i = 1, \tag{6}$$

$$M_0(a_i) - \sum_{l \in \mathbb{W}} \gamma(a_i, y_l) \cdot M(y_l) - \sum_{j \in \mathbb{V}} z_{ai}^{y_j} \cdot M(y_j) \leq \gamma(b_{i-1n_{i-1}}, a_i) - 1 + Q \cdot (1 - d_{i-1}), \forall i \in \{2, 3, \ldots, m\}, \tag{7}$$

where Q is a positive constant that must be sufficiently large. Eqs. (6) and (7) claim that if $d_i = 1$, the corresponding constraint is active, and if $d_i = 0$, the corresponding constraint always holds since Q is big enough. Hence, we only need to guarantee that $d_i = 1$ if $M(b_{in_i}) > 0$ and $d_i = 0$ if $M(b_{in_i}) = 0$, i.e.,

$$d_i \leq M(b_{in_i}) \cdot Q, \forall i \in \{1, 2, \ldots, m\}, \tag{8}$$

$$M(b_{in_i}) \leq d_i \cdot Q, \forall i \in \{1, 2, \ldots, m\}. \tag{9}$$

By Eqs. (6)–(9), we ensure that the resources held by operation place $b_{i-1n_{i-1}}$ (b_{mn_m}), $i \in \{2, 3, \ldots, m\}$, cannot be released by requesting tokens from a_i (a_1). Nevertheless, we need to guarantee that operation place $b_{i-1n_{i-1}}$ (b_{mn_m}) cannot be released by other resources by Definition 4. Let Δ_{i-1} be the alternative release set of $(b_{i-1n_{i-1}}, a_i)$ with $i \in \{2, 3, \ldots, m\}$ and Δ_m the alternative release set of (b_{mn_m}, a_1). We have

$$\sum_{l \in \mathbb{W}'} \gamma(a_k, b_l) \cdot M(b_l) + \sum_{j \in \mathbb{V}'} z_{a_k}^{b_j} \cdot M(b_j) \leq M_0(a_k), \quad \forall k \in \mathbb{A}, \forall i \in \{1, 2, \ldots, m\}, \tag{10}$$

$$M(b_l) \leq \min\{\lfloor \frac{1}{\gamma(a_s, b_l)} \cdot M_0(a_s) \rfloor, \lfloor \frac{1}{z_{a_t}^{b_l}} \cdot M_0(a_t) \rfloor \mid (a_s, b_l) \in E, a_s \neq a_k, z_{a_t}^{b_l} > 0\} \quad \forall l \in \mathbb{W}, \forall k \in \mathbb{A}, \forall i \in \{1, 2, \ldots, m\}, \tag{11}$$

$$M(b_j) \leq \min\{\lfloor \frac{1}{\gamma(a'_s, b_j)} \cdot M_0(a'_s) \rfloor, \lfloor \frac{1}{z_{a'_t}^{b_j}} \cdot M_0(a'_t) \rfloor \mid (a'_s, b_j) \in E, z_{a'_t}^{y_j} > 0, a_t \neq a_k\} \quad \forall l \in \mathbb{V}, \forall k \in \mathbb{A}, \forall i \in \{1, 2, \ldots, m\}, \tag{12}$$

$$M_0(a_k) - \sum_{l \in \mathbb{W}'} \gamma(a_k, b_l) \cdot M(b_l) - \sum_{j \in \mathbb{V}} z_{a_k}^{b_j} \cdot M(b_j) \leq \gamma(b_{in_i}, a_k) - 1 + Q \cdot (1 - d_i), \forall q \in \mathbb{A}, \forall i \in \{1, 2, \ldots, m\}, \tag{13}$$

where $\mathbb{W}' = \{l | (a_k, b_l) \in E_{rd}\}$, $\mathbb{V}' = \{j | z_{a_k}^{b_j} > 0\}$, and $\mathbb{A} = \{k | (b_{in_i}, a_k) \in \Delta_i\}$ are three index sets and $M_0(a_k)$ is the initial marking of resource place a_k. To avoid any confusion, we clarify Eqs. (10)–(13) in the following example.

Consider the Petri net in Fig. 6(a). Its resource requirement graph is depicted in Fig. 6(b) by removing nodes and arcs that do not appear in any resource-operation loop. There are two resource-operation loops, which are $\beta_1 = p_6 \to (p_1) \to p_7 \to (p_4) \to p_6$ and $\beta_2 = p_6 \to (p_1) \to p_8 \to (p_4) \to p_6$. Let us

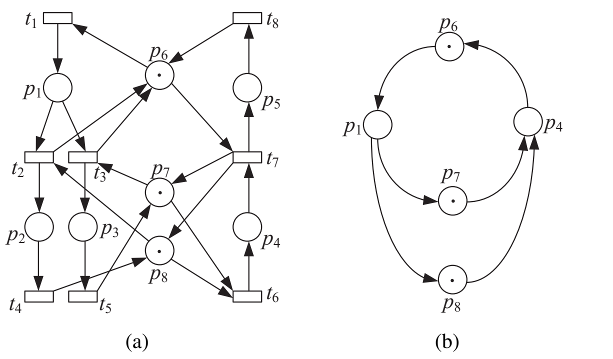


Fig. 6: (a) Petri net model, (b) resource requirement graph.

discuss β_1 to clarify Eqs. (10)–(13). As seen, the alternative release set of (p_1, p_7) is $\Delta_1 = \{(p_1, p_8)\}$ and the alternative release set Δ_2 of (p_4, p_6) is empty, which implies that the resources in p_1 can be released by requesting resources from p_7 or p_8 and the resources in p_4 can be released by only requesting resources from p_6. Therefore, for p_1, we should not only guarantee that the resources in p_1 cannot be released by p_7, but also by p_8. Given that we already guarantee that the resources in p_1 cannot be released by requesting resources from p_7 by Eqs. (1)–(3) and (6)–(9), we have $M(p_4) \leq 1$ for p_8 by Eq. (10). Since p_4 only needs resource from p_8, we do not need Eqs. (11) and (12). In addition, by Eq. (13), we have $1 - M(p_4) \leq 0$. Thus, once p_1 gets a resource unit from p_6, it cannot be released by either p_7 or p_8.

By Eqs. (6), (7), and (13), the resources held by operation places b_{in_i} ($i \in \{1, 2, \ldots, m\}$) cannot be released by any resource place. At the same time, we have to guarantee that other operation places in β, besides b_{in_i}, cannot be released either. In the interest of simplicity, we only consider a situation where there are arcs from the mentioned operation places to the resource places in β. Let set_β be the set of all the places that appear in β. We have

$$M_0(a_i) - \sum_{l \in \mathbb{W}} \gamma(a_i, y_l) \cdot M(y_l) - \sum_{j \in \mathbb{V}} z_{ai}^{y_j} \cdot M(y_j) \leq \gamma(y_u, a_i) - 1 + Q \cdot (1 - d_u), \forall u \in \mathbb{H}, \tag{14}$$

$$d_u \leq M(y_u) \cdot Q, \forall u \in \mathbb{H}, \tag{15}$$

$$M(y_u) \leq d_u \cdot Q, \forall u \in \mathbb{H}, \tag{16}$$

where $\mathbb{H} = \{u | (y_u, a_i) \in E, y_u = set_\beta \cap P_A, a_i = set_\beta \cap P_R\}$ is an index set. Moreover, according to Definition 3, the operation places in the release-free set $\mathcal{R}_\mathrm{f}$ can release the resources that they currently hold, without further requesting any resource. In other words, the operation places in $\mathcal{R}_\mathrm{f}$ do not participate in the resource competition. For each operation place $y_o \in \mathcal{R}_\mathrm{f} \cap set_\beta$, we have:

$$M(y_o) = 0, \forall y_o \in \mathcal{R}_\mathrm{f} \cap set_\beta. \tag{17}$$

For an operation pattern $\delta_k = (b_{k1}, b_{k2}, \ldots, b_{kn_k}) \in \beta$, consider a situation where an operation place b_{ki} (representing a processing stage) requires more resource units for every type of resources than the next processing stage represented by

b_{ki+1} ($i \in \{1,2,\ldots,n_k-1\}$). In this case, the resources held by b_{ki} can move to b_{ki+1} without requiring resources. To guarantee that no resources held by the operation places in β are capable of being released, we have:

$$M(b_{ki}) = 0, \forall b_{ki} \in \mathbb{B}, \forall k \in \{1,2,\ldots,m\}, \tag{18}$$

where $B = \{b_{ki} | \forall a \in P_R : z_a^{b_{ki}} \geq z_a^{b_{ki+1}} \vee \gamma(a, b_{ki}) \geq z_a^{b_{ki+1}}, i \in \{1,2,\ldots,n_k-1\}\}$. Let us consider again the resource requirement graph depicted in Fig. 5(b). According to the resource requirement graph, we know that $z_{p_3}^{p_2} = 1$ and $\gamma(p_3, p_1) = 2$. In words, the tokens in p_1 can move into p_2 without requesting any resources thanks to $\gamma(p_3, p_1) \geq z_{p_3}^{p_2}$. Hence, we have $M(p_1) = 0$. In addition, our objective is to identify pre-partial deadlocks, where a specific quantity of tokens held by the operation places in β cannot be released. In other words, there is at least one operation place that holds resources, which leads to

$$\sum_{k=1}^{m} \sum_{i=1}^{n_k} M(b_{ki}) \geq 1. \tag{19}$$

Given a resource-operation loop β, Eqs. (1)–(3) and (6)–(19) are grouped into a set of linear inequalities, namely the potential pre-partial deadlock calculation (PPDC). The markings associated with the PPDC of β are named potential pre-partial deadlocks. The feasibility of PPDC can also be solved by linking an objective function to all constraints and solving the obtained ILPP.

Let $||\beta||$ denote the set of all operation places that appear in the aforementioned inequalities of resource-operation loop β. According to Eqs. (1) and (10), not only do the operation places of β appear in the inequalities, but also the operation places of other resource-operation loops may appear in the inequalities. Therefore, the set $||\beta||$ can be divided into two partitions: $||\beta||^o$ and $||\beta||^e$, where $||\beta||^o$ is a set containing the operation places of β and $||\beta||^e$ is a set containing those not of β. In addition, by solving the PPDC of β, we can only ensure that the tokens held by the operation places in $||\beta||^o$ cannot be released. It is unknown whether the tokens held by the operation places in $||\beta||^e$ can be released. In this case, the obtained potential pre-partial deadlocks need to be further checked to decide whether one is an actual pre-partial deadlock. Algorithm 1 is used to determine the actual pre-partial deadlocks.

Let's briefly go over how Algorithm 1 is implemented through an example. Consider again the resource requirement graph G_1 in Fig. 4. By solving the PPDC of each resource-operation loop in G_1, we have the potential pre-partial deadlock sets: $\mathcal{S}_{\beta_1} = \{2p_2+p_7, 3p_7, p_2+2p_7\}$, $\mathcal{S}_{\beta_2} = \{3p_7, 2p_2+p_7, p_2+2p_7, 2p_2+p_4+p_7, p_4+3p_7, 3p_2+p_4, p_2+p_4+2p_7\}$, and $\mathcal{S}_{\beta_3} = \{3p_2+p_4, 2p_2+p_4+p_7, p_4+3p_7, p_2+p_4+2p_7\}$. Let us first discuss the potential deadlock marking $M_1 = 3p_7 \in \mathcal{S}_{\beta_1}$. By the resource requirement graph G_1, we have $||\beta_1||^o = \{p_7\}$ and $||\beta_1||^e = \{p_2, p_8\}$. It is obvious that both $M_1(p_2)$ and $M_1(p_8)$ are equal to zero. Therefore, $M_1 = 3p_7$ is an actual pre-partial deadlock. Furthermore, let us discuss in detail how to check whether the marking $M_2 = 2p_2 + p_7 \in \mathcal{S}_{\beta_1}$ is an actual pre-partial deadlock. By $M_2(p_2) = 2 \neq 0$, we need to check if $2p_2$ is a part of a potential pre-partial deadlock in other resource-operation loops. In this case, we have $||\beta_3||^o = \{p_2, p_3, p_4\}$ and $||\beta_3||^e = \{p_7, p_8\}$. It is noted that $||\beta_3||^o \cap ||\beta_1||^e = \{p_2\} \neq \emptyset$ and $||\beta_1|| \cap ||\beta_3|| = \{p_2, p_7, p_8\}$. There is a marking $M_3 = 2p_2 + p_4 + p_7 \in \mathcal{S}_{\beta_3}$ such that $M_2(p_2) = M_3(p_2) = 2$, $M_2(p_7) = M_3(p_7) = 1$, and $M_2(p_8) = M_3(p_8) = 0$. Furthermore, we have $||\beta_1||^e \cup ||\beta_3||^e \setminus (||\beta_1||^o \cup ||\beta_3||^o) = \{p_8\}$, i.e., there exists an operation place p_8, which is neither in $||\beta_1||^o$ nor in $||\beta_3||^o$. However, we have

Algorithm 1 Computation of pre-partial deadlocks

```
Input: A Petri net system $(N, M_0)$ with $N = (P^0 \dot\cup P_A \dot\cup P_R, T, F, W)$.
Output: A set of pre-partial deadlocks $\mathcal{M}_{\text{PPD}}$.
1: $\mathcal{M}_{\text{PPD}} := \emptyset$, $Temp := \emptyset$.
2: Construct a resource requirement graph $G = (P_G, \chi, E, \gamma, S, \phi)$ of $N$.
3: Compute the set of resource-operation loops $\mathcal{N}_\beta$ by Definitions 5 and 6.
4: for all $\beta \in \mathcal{N}_\beta$ do
5:   Design the PPDC of $\beta$ and solve it.
6:   Let $\mathcal{S}_\beta$ denote the set of the obtained potential pre-partial deadlocks by solving the PPDC of $\beta$.
7: end for
8: for all $\beta \in \mathcal{N}_\beta$ do
9:   for all $M \in \mathcal{S}_\beta$ do
10:    if $||\beta||^e = \emptyset$ or $\forall p \in ||\beta||^e$, $M(p) == 0$ then
11:      $\mathcal{M}_{\text{PPD}} := \mathcal{M}_{\text{PPD}} \cup \{M\}$.
12:    else
13:      $||\beta^*|| = ||\beta||$, $M^* = M$.
14:      for all $\beta' \in \mathcal{N}_\beta$ such that $||\beta'||^o \cap ||\beta^*||^e \neq \emptyset$ do
15:        if $\exists M' \in \mathcal{S}_\beta$ such that $\forall p \in ||\beta'|| \cap ||\beta^*||$, $M'(p) == M^*(p)$ then
16:          for all $M'$ do
17:            $M^* := \sum_{p \in ||\beta^*||} M^*(p)p + \sum_{p \in ||\beta'|| \setminus ||\beta^*||} M'(p)p$.
18:            $||\beta^*||^e == ||\beta^*||^e \cup ||\beta'||^e \setminus (||\beta^*||^o \cup ||\beta'||^o)$.
19:            $||\beta^*|| := ||\beta^*|| \cup ||\beta'||$.
20:            $Temp := Temp \cup \{M^*, M'\}$.
21:            if $||\beta^*||^e == \emptyset$ or $\forall p \in ||\beta^*||^e$, $M^*(p) = 0$ then
22:              $\mathcal{M}_{\text{PPD}} := \mathcal{M}_{\text{PPD}} \cup Temp$.
23:            else
24:              Go to step 14.
25:            end if
26:          end for
27:        else
28:          $Temp := \emptyset$.
29:          Break.
30:        end if
31:      end for
32:    end if
33:  end for
34: end for
35: Output a set of pre-partial deadlocks $\mathcal{M}_{\text{PPD}}$.
```

$M_2(p_8) = M_3(p_8) = 0$. As a result, both $M_2 = 2p_2 + p_7$ and $M_3 = 2p_2 + p_4 + p_7$ are pre-partial deadlocks.

Definition 8: Given a Petri net system (N, M_0) and a pre-partial deadlock M^* of a resource-operation loop β, the coset of M^* is defined as $[M^*] = \{M \in R(N, M_0) | \forall p \in ||\beta|| : M(p) = M^*(p)\}$.

Theorem 1: A marking $M \in R(N, M_0)$ is a pre-partial deadlock marking iff there exists a pre-partial deadlock $M^* \in \mathcal{M}_{\text{PPD}}$ such that $M \in [M^*]$.

Proof: ($\Leftarrow$) We aim to show that if there exists a pre-partial deadlock $M^* \in \mathcal{M}_{\text{PPD}}$ such that a marking $M \in R(N, M_0)$ is in the coset of M^*, then M is a pre-partial deadlock marking. By Eqs. (1) and (10), the system resources are allocated to the corresponding operation places by the property of resource conservativeness, which will lead to a situation where certain operation places can hold resources. In other words, the input transitions of the operation places are enabled to get the processing stages represented by the operation places started by the firing rules of Petri nets. To finish the processes, system resources are required, i.e., the output transitions of the operation places need resources to be enabled. However, by Eqs. (6), (7), (13), and (14), the amount of remaining resources after resource allocation is not big enough to finish the processes, which means that the output transitions of the operation places cannot be enabled. Nonetheless, it is possible that the remaining resources are still enough to start the processes, which can only decrease the amount of the remaining resources. The output transitions will never be enabled anymore. Resource allocation will stop when the amount of resources is not enough anymore to start the processes. That is to say, all the input and the output transitions cannot be enabled eventually. According to Definition 7, for any reachable marking $M \in [M^*]$, M is a pre-partial deadlock marking.

($\Rightarrow$) We aim to show that if a marking $M \in R(N, M_0)$ is a pre-partial deadlock marking, then there exists a pre-partial deadlock $M^* \in \mathcal{M}_{\text{PPD}}$ such that $M \in [M^*]$. First, we prove that there is a group of operation places satisfying Eqs. (6), (7), (13), and (14) at marking M. By contradiction, assume that there is not a group of operation places that satisfy Eqs. (6), (7), (13), and (14), which indicates that all operation places are capable of releasing the resources that they are holding. The ouput transitions of the operation places can always be enabled, which contradicts our assumption. In addition, all reachable markings satisfy the property of resource conservativeness, i.e., there is a group of operation places satisfying Eqs. (1) and (10) at marking M. Therefore, there is a pre-partial deadlock $M^* \in \mathcal{M}_{\text{PPD}}$ such that $M \in [M^*]$. ■

IV. Deadlock Resolution

In this section, a new notion called a pre-partial deadlock set/recovery transition instance (DTI) is proposed. A DTI is a pair of the form $(\mathcal{M}_d, t_r)$ consisting of a set of pre-partial deadlocks $\mathcal{M}_d$ and a recovery transition t_r that is designed to be enabled at every marking in $\mathcal{M}_d$. By adding DTIs, the controlled net model is live and preserves all reachable markings. We now introduce the definitions that are useful for designing DTIs.

Definition 9: Given a pre-partial deadlock M, the support of M is defined as $||M|| = \{p \in P_A | M(p) \neq 0\}$.

Definition 10: Given a resource requirement graph $G = (P_G, \chi, E, \gamma, S, \phi)$ and a pre-partial deadlock M, a path of M is defined as a sequence of operation places: $\rho = b_1 b_2 \dots b_n$, satisfying:

- for all $i \in \{1, 2, \dots, n-1\}$, there exists a resource place a_i such that $(b_i, a_i) \in E$ and $(a_i, b_{i+1}) \in E$,
- for all $i \in \{1, 2, \dots, n\}$, $b_i \in ||M||$.

A path ρ is called a circle if there exists a resource place a' such that $(b_n, a') \in E$ and $(a', b_1) \in E$.

Definition 11: Given a path $\rho = b_1 b_2 \dots b_n$, the support of ρ is defined as $||\rho|| = \{b_i | i \in \{1, 2, \dots, n\}\}$.

Definition 12: Given a pre-partial deadlock M, a path set of M is defined as $C_M = \{\rho_1, \rho_1, \dots, \rho_n\}$, such that:

- $||\rho_1|| + ||\rho_2|| + \cdots + ||\rho_n|| = ||M||$,
- for all $i, j \in \{1, 2, \dots, n\}$, $||\rho_i|| \cap ||\rho_j|| = \emptyset$ if $i \neq j$,

where ρ_i (ρ_j) is a path of M. C_M is called a minimal path set of M if the number of the paths in C_M is no greater than any other path set of M.

Consider Petri net N_1 in Fig. 3. The resource requirement graph of N_1 is depicted in Fig. 4. By Algorithm 1, the set of pre-partial deadlocks of Petri net N_1 is $\mathcal{M}_{\text{PPD}} = \{3p_7, p_4 + 3p_7, 3p_2 + p_4, 2p_2 + p_7, 2p_2 + p_4 + p_7, p_2 + 2p_7, p_2 + p_4 + 2p_7\}$. For example, the support of pre-partial deadlock $M = p_4 + 3p_7$ is $\{p_4, p_7\}$. There are three paths of M: $\rho_{M_1} = p_4$, $\rho_{M_2} = p_7$, and $\rho_{M_3} = p_4 p_7$. The minimal path set of M is $C_M = \{p_4 p_7\}$. In addition, none of the paths of M is a circle.

Let $\rho = b_1 b_2, \dots, b_n$ be a path of a pre-partial deadlock M. According to Definition 10, there is a resource place a_i such that $(b_i, a_i) \in E$ and $(a_i, b_{i+1}) \in E$ for all $i \in \{1, 2, \dots, n-1\}$. For the sake of intuition, we denote the path as $\rho = b_1(a_1)b_2(a_2), \dots, b_{n-1}(a_{n-1})b_n$. Since M is a pre-partial deadlock by Algorithm 1, none of the resources held by the operation places in $||\rho||$ can be released. Specifically, operation place b_1 requires the resource in resource place a_1 that is currently held by b_2, b_2 requires the resource from a_3 that is currently held by b_3, and so on. Finally, b_{n-1} requires the resource from a_{n-1} that is currently held by b_n. Obviously, each operation place in $||\rho||$ will be able to release the resource if b_n can return the resource back to a_{n-1}. Thus, a recovery transition needs to be designed for b_n to force it to return the resource back. In case that ρ is a circle, we can design a recovery transition for any one of the operation places in ρ. To this end, we define the new notion of a recovery set for a pre-partial deadlock.

Definition 13: Given a pre-partial deadlock M and its minimal path set $C_M = \{\rho_1, \rho_2, \dots, \rho_m\}$, a recovery set of M is defined as:

$$r_i = \begin{cases} \{b_{i1}, b_{i2}, \dots, b_{in}\}, & \text{if } \rho_i \text{ is a circle;} \\ \{b_{in}\}, & \text{otherwise,} \end{cases}$$

where $\rho_i = b_{i1} b_{i2} \dots b_{in}$ ($i \in \{1, 2, \dots, m\}$) is a path in C_M.

In Definition 13, a recovery set r_i of a pre-partial deadlock M consists all operation places in a path $\rho_i \in C_M$ if ρ_i is a circle. Otherwise, a recovery set r_i only consists of the last operation place in ρ_i, i.e., b_{in}. Hence, in order to recover M, we need to select one operation place from every recovery set of M and design a recovery transition for each selected place. Let us consider again the pre-partial deadlock $M = p_4+3p_7$ of N_1. There is only one recovery set of M, which is $r = \{p_7\}$, since the minimal path set of M only has one path that is p_4p_7.

Next, an approach is presented to minimize the number of the selected operation places, when designing recovery transitions. Let R_M denote the set of all the recovery sets of a pre-partial deadlock M. For a pre-partial deadlock M, a binary variable $h_j \in \{0, 1\}$ is assigned for each operation place of each recovery set to represent whether p_j is selected. For a recovery set $r \in R$, we have

$$h_j = \begin{cases} 1, & \text{if } p_j \in r \text{ is selected;} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Let $\mathcal{M}_{\text{PPD}}$ be the set of all pre-partial deadlocks. To ensure that at least one operation place is selected from each recovery set of every pre-partial deadlock, we have

$$\sum_{p_j\in r} h_j \geq 1, \forall M \in \mathcal{M}_{\text{PPD}}, \forall r \in R_M, \tag{21}$$
$$h_j \in \{0, 1\}, \forall p_j \in r, \forall M \in \mathcal{M}_{\text{PPD}}, \forall r \in R_M.$$

In this scenario, our objective function aims to minimize the quantity of selected operation places.

$$\min\{\sum_{M\in\mathcal{M}_{\text{PPD}}} \sum_{r\in R_M} \sum_{p_j\in r} h_j\}.$$

Thus, we have an ILPP, namely the minimal universal set problem (MUSP):

$$\begin{aligned} \text{MUSP: min} \quad & \{\sum_{M\in\mathcal{M}_{\text{PPD}}} \sum_{r\in R_M} \sum_{p_j\in r} h_j\} \\ \text{s.t.} \quad & \sum_{p_j\in r} h_j \geq 1, \forall r \in R_M, \forall M \in \mathcal{M}_{\text{PPD}}, \\ & h_j \in \{0, 1\}, \forall p_j \in r, \forall r \in R_M, \forall M \in \mathcal{M}_{\text{PPD}}. \end{aligned} \tag{22}$$

Let U denote the minimal universal set consisting of all the selected operation places. The objective function h can minimize the number of the selected operation places of all recovery sets. If $h_j = 1$, operation place p_j is selected and covered in U. If $h_j = 0$, operation place p_j is not selected and not covered by U. In this case, it is necessary to develop a recovery transition for each operation place in U. The details of designing recovery transitions are stated as follows.

Let (N, M_0) be a net system and (N_r, M_0) be the controlled net system after adding a set of recovery transitions. A transition t_r, with incidence vector $[N_r](P, t_r) = [x_1, x_2, x_3, \ldots, x_n]^T$, is designed for an operation place $p \in U$, where $n = |P|$. By firing t_r, the resources held by p should be released. Hence, we have

$$[N_r](p, t_r) = -1. \tag{23}$$

By firing t_r, the resources released from p should return to the corresponding resource places. As discussed earlier, the processing stage, which is represented by p, needs $\gamma(p_e, p)$ amount of resource units from resource place p_e if there is an arc from p_e to p in the resource requirement graph. Furthermore, the processing stage needs $z^p_{p_s}$ resource units from resource place p_s if $z^p_{p_s} > 0$. Let $A = \{p_e \in P_R|(p_e, p) \in E\}$ and $B = \{p_s \in P_R|z^p_{p_s} > 0\}$ be two sets of resource places. Thus, we have

$$[N_r](p_e, t_r) = \gamma(p_e, p), \forall p_e \in A, \tag{24}$$

$$[N_r](p_s, t_r) = z^p_{p_s}, \forall p_s \in B. \tag{25}$$

For the remaining operation places and resource places, the numbers of tokens of the places should not change when firing the recovery transition t_r, i.e.,

$$[N_r](p_l, t_r) = 0, \forall p_l \in P \setminus (P^0 \cup \ A \cup B \cup \{p\}). \tag{26}$$

By Eqs. (23)–(26), we know that, by firing t_r, the tokens in p decrease by one. Moreover, the quantity of the tokens returned to the corresponding resource place is exactly the same as the number of the tokens required by operation place p from the resource place. The amount of the tokens in the remaining operation places and resource places stays unchanged. To finalize the incidence vector of t_r, we must calculate x_i's for all $p_{0i} \in P^0$. By adding the recovery transition t_r to the original net (N, M_0), the conservativeness of the controlled net system (N_r, M_0) must be guaranteed. According to our previous work [25], we have

$$x_i = -I_{p_{0i}}(p)x_p/I_{p_{0i}}(p_{0i}), \forall p_{0i} \in P^0. \tag{27}$$

To this end, the incidence vector $[N_r](P, t_r)$ associated with t_r is precisely established. Note that recovery transition t_r will be enabled and return the resources to the corresponding resource places and idle places, whenever there are resources in the operation place p. This situation may hinder the system from proceeding to the next step of the processing stage represented by p, which means that the system will not be able to operate properly. Therefore, we need to ensure that the recovery transition is only allowed to be enabled if the system enters one of the pre-partial deadlock states. We now introduce the notion of a pre-partial deadlock set/recovery transition instance (DTI).

Definition 14: A pre-partial deadlock set/recovery transition instance (DTI), denoted as $(\mathcal{M}_d, t_r)$, is a pair of a set of pre-partial deadlocks $\mathcal{M}_d$ and a recovery transition t_r that is only allowed to be enabled at each marking $M \in \mathcal{M}_d$.

Definition 15: Let $(\mathcal{M}_d, t_r)$ be a DTI and t_r a recovery transition that is designed for operation place p. A pre-partial deadlock $M \in \mathcal{M}_d$ if there exists a recovery set $r \in R$ such that $p \in r$, where R is the set of all the recovery sets of M.

Consider the pre-partial deadlock $M = p_4 + 3p_7$ of Petri net N_1 depicted in Fig. 3. As discussed earlier, there is only one recovery set of M, which is $r = \{p_7\}$. Hence, we need to design a recovery transition for p_7 in order to recover M. Suppose that t_r is a recovery transition designed for p_7. Transition t_r is not necessarily enabled simply because there are tokens in p_7. Transition t_r should be enabled at M. In other words, transition t_r is enabled when there is one token in p_4 and three tokens in p_7 observed in the system. We present Algorithm 2 to generate DTIs for a given Petri net system.

Algorithm 2 Generation of DTIs for a Petri net system

```
Input: A Petri net system $(N, M_0)$.
Output: A live Petri net system $(N_r, M_0)$ by adding DTIs.
 1: $\Omega := \emptyset$.
 2: Compute the set of all the pre-partial deadlocks by Algorithm1, denoted as $\mathcal{M}_{\text{PPD}}$.
 3: for all $M \in \mathcal{M}_{\text{PPD}}$ do
 4:     Compute a minimal path set of $M$ by Definition 12, denoted as $C_M$.
 5:     for all $\rho \in C_M$ do
 6:         Compute the set of all the recovery sets of $M$, denoted as $R$.
 7:     end for
 8: end for
 9: Design MUSP and solve it.
10: Let $U$ be the minimal universal set that covers the selected operation places.
11: for all $p \in U$ do
12:     Calculate the incidence vector of $t_r$ by Eqs. (23)–(27).
13:     Compute the DTI with respect to $t_r$ by Definitions 14 and 15, denoted as $(\mathcal{M}_d, t_r)$.
14:     $\Omega = \Omega \cup \{(\mathcal{M}_d, t_r)\}$.
15: end for
16: Add all the DTIs to the original Petri net and denote the controlled net system as $(N_r, M_0)$.
17: Output $(N_r, M_0)$.
```

Theorem 2: The controlled net system (N_r, M_0) obtained after executing Algorithm 2 is live with all reachable markings.

Proof: Let $\mathcal{M}_{\text{PPD}}$ be the set of the pre-partial deadlocks obtained by Algorithm 1 and $M_1 \in R(N, M_0)$ be a pre-partial deadlock marking. According to Theorem 1, there exists a pre-partial deadlock $M_1^* \in \mathcal{M}_{\text{PPD}}$ such that $M_1 \in [M_1^*]$. By Algorithm 2, a recovery transition is developed to be enabled at M_1. In other words, there exists a recovery transition that can be enabled at every pre-partial deadlock marking. More importantly, there exist a sequence of recovery transitions $\sigma = t_{r1}, t_{r2}, \ldots, t_{rn-1}$ and pre-partial deadlock markings M_1, M_2, …, M_{n-1} such that firing σ at M_1, which is denoted as $M_1[t_{r1}\rangle M_2[t_{r2}\rangle, \ldots, M_{n-1}[t_{rn-1}\rangle M_n$, can eventually yield a new marking M_n that is not a pre-partial deadlock marking. According to Definition 7, there does not exist a transition t from the original Petri net such that firing t at M_n can go back to any pre-partial deadlock marking, since M_n is not a pre-partial deadlock marking. No recovery transitions can be enabled at a marking that is not a pre-partial deadlock. In other words, there is no new livelocks formed in the controlled system. Moreover, it is trivial that the number of reachable states of the Petri net will not change by adding the recovery transitions. Therefore, all pre-partial deadlock markings are recovered into markings that are not pre-partial deadlock markings.

Suppose that there is a deadlock M in (N_r, M_0). According to Proposition 1 in [25] (any deadlock is a partial deadlock) and Definition 7, M is also a pre-partial deadlock marking. As discussed, all the pre-partial deadlock markings are recovered by DTIs. Thus, M is not a deadlock marking. Furthermore, suppose that (N_r, M_0) is deadlock-free (livelock), which means that for all $M' \in R(N_r, M_0)$, there exists $t \in T$, $M'[t\rangle$. In other words, there exists a nonempty subset $T' \subseteq T$ satisfying for all $M' \in R(N, M)$, there does not exist a transition $t \in T'$ such that $M'[t\rangle$. According to Definition 7, M' is a pre-partial deadlock marking. However, there is no longer pre-partial deadlock markings in (N_r, M_0). Hence, the controlled net system (N_r, M_0) is live and preserves all reachable markings. ■

We provide the details of the computation of pre-partial deadlocks and the design of DTIs by the proposed approach for Petri net N_1 in Fig. 3. There are three resource-operation loops in the resource requirement graph in Fig. 4, which are $\beta_1 = p_{11} \to (p_7) \to p_{11}$, $\beta_2 = p_{11} \to (p_7, p_8) \to p_{12} \to (p_4) \to p_{11}$, and $\beta_3 = p_{11} \to (p_2) \to p_{13} \to (p_3) \to p_{12} \to (p_4) \to p_{11}$. For β_1, there is only one resource place, i.e., p_{11}. Let x_i denote the marking of place p_i. We have $x_2+x_7+2x_8 \leq 3$. Since the processing stages, represented by operation place p_2, p_7, and p_8, only need resources from p_{11}, Eqs. (2) and (3) are omitted. By Eqs. (6)–(9), we have $3 - x_2 - x_7 - 2x_8 \leq 1 - 1 + Q \cdot (1 - d_7)$ with $d_7 \leq x_7 \cdot Q$ and $x_7 \leq d_7 \cdot Q$. In addition, there is only one arc that starts from an operation place to a resource place, which is (p_7, p_{11}). By Definition 4, the alternative release set of (p_7, p_{11}) is empty. Therefore, we do not have Eqs. (10)–(16). Moreover, we have $x_8 = 0$ because p_8 is a release-free operation place. Eq. (18) is also omitted since there is only one operation place in β_1. Finally, we have $x_7 \geq 1$ by Eq. (19). All the constraints of β_1 are illustrated as: $x_2 + x_7 + 2x_8 \leq 3$, $3 - x_2 - x_7 - 2x_8 \leq Q \cdot (1 - d_7)$, $d_7 \leq x_7 \cdot Q$, $x_7 \leq d_7 \cdot Q$, $x_8 = 0$, and $x_7 \geq 1$.

For β_2, since $x_8 = 0$, there is no need to disable (p_8, p_{12}). We have the following constraints: $x_2+x_7+2x_8 \leq 3$, $x_4 \leq 1$, $3-x_2-x_7-2x_8 \leq Q\cdot(1-d_4)$, $3-x_2-x_7-2x_8 \leq Q\cdot(1-d_7)$, $x_8 = 0$, $x_4 + x_7 + x_8 \geq 1$, $d_i \leq x_i \cdot Q$, $x_i \leq d_i \cdot Q$, and $i \in \{4, 7\}$. Similarly, for β_3, we have the following constraints: $x_2+x_7+2x_8 \leq 3$, $x_3+x_4 \leq 1$, $x_4 \leq 1$, $3-x_2-x_7-2x_8 \leq Q\cdot(1-d_4)$, $1-x_4 \leq Q\cdot(1-d_3)$, $1-x_3-x_4 \leq Q\cdot(1-d_2)$, $x_8 = 0$, $x_2 + x_3 + x_4 \geq 1$, $d_i \leq x_i \cdot Q$, $x_i \leq d_i \cdot Q$, and $i \in \{2, 3, 4\}$.

By applying Algorithm 1, we have the set of all pre-partial deadlocks, i.e., $\mathcal{M}_{\text{PPD}} = \{3p_7, p_4 + 3p_7, 3p_2 + p_4, 2p_2 + p_7, 2p_2+p_4+p_7, p_2+2p_7, p_2+p_4+2p_7\}$. Moreover, the deadlock markings of the original Petri net are $6p_1+3p_2+p_4+10p_6$, $7p_1+2p_2+p_4+9p_6+p_7$, $8p_1+p_2+p_4+8p_6+2p_7$, $9p_1+p_4+7p_6+3p_7$ and $10p_1+7p_6+3p_7+p_{12}+p_{13}$. Note that for each deadlock marking M of the original Petri net, there exists a pre-partial deadlock $M^* \in \mathcal{M}_{\text{PPD}}$ such that $M \in [M^*]$. The details of each pre-partial deadlock are summarized in Table I including the minimal path sets and the recovery sets. The minimal universal set is $U = \{p_2, p_7\}$ by solving the MUSP. Let t_{r1} be the recovery transition for p_2 with $[N_r](P, t_{r1}) = [x_1, x_2, \ldots, x_{13}]^T$. By Eqs. (23) and (24), we have $x_2 = -1$ and $x_{11} = 1$. Furthermore, there are two minimal P-semiflows, which are $I_1 = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]^T$ and $I_2 = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0]^T$. By Eq. (27), we have $x_1 = 1$ and $x_6 = 0$. The complete incidence vector of t_{r1} is $[N_r](P, t_{r1}) = [1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]^T$.

TABLE I: Details of pre-patial deadlocks

pre-partial deadlock	minimal path set	recovery set
$3p_7$	$\{p_7\}$	$\{p_7\}$
$p_4 + 3p_7$	$\{p_4p_7\}$	$\{p_7\}$
$3p_2 + p_4$	$\{p_4p_2\}$	$\{p_2\}$
$2p_2 + p_7$	$\{p_7p_2\}$	$\{p_2\}$
$2p_2 + p_4 + p_7$	$\{p_4p_7p_2\}$	$\{p_2\}$
$p_2 + 2p_7$	$\{p_7p_2\}$	$\{p_2\}$
$p_2 + p_4 + 2p_7$	$\{p_4p_7p_2\}$	$\{p_2\}$

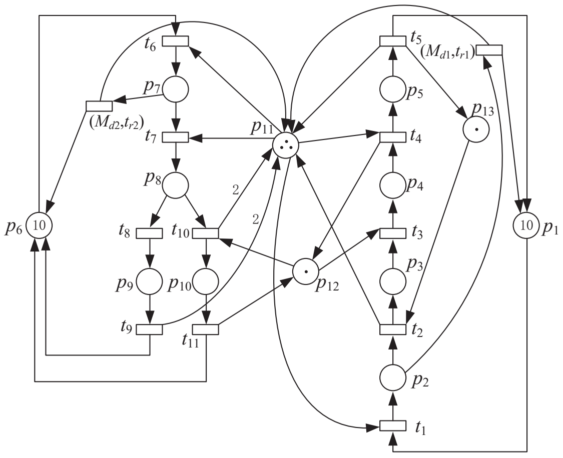


Fig. 7: The controlled net system (N_{r1}, M_{01}).

Similarly, let t_{r2} be the recovery transition for p_7. The complete incidence vector of t_{r2} is expressed as $[N_r](P, t_{r2}) = [0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 1, 0, 0]^T$. The DTI with respect to t_{r1} (t_{r2}) is denoted as $(\mathcal{M}_{d1}, t_{r1})$ ($(\mathcal{M}_{d2}, t_{r2})$). The details of the two DTIs are summarized in Table II. The original Petri net in Fig. 3 has 96 reachable markings. After adding the two DTIs, the controlled net system, which is depicted in Fig. 7, is live with the same 96 markings.

TABLE II: Details of the two DTIs

$(\mathcal{M}_{di}, t_{ri})$	$^\bullet t_{ri}$	$t_{ri}{}^\bullet$	$\mathcal{M}_{di}$
$(\mathcal{M}_{d1}, t_{r1})$	p_2	p_1, p_{11}	$\{3p_2 + p_4,\ 2p_2 + p_7,\ 2p_2 + p_4 + p_7,\ p_2 + 2p_7,\ p_2 + p_4 + 2p_7\}$
$(\mathcal{M}_{d2}, t_{r2})$	p_7	p_6, p_{11}	$\{3p_7,\ p_4 + 3p_7\}$

We next discuss the computational complexity of the proposed method. First, a resource requirement graph is constructed following Definition 2, which is of polynomial complexity. Second, we need to solve ILPPs to compute pre-partial deadlocks, which is in the worst case exponential. Furthermore, we must solve another ILPP to design recovery transitions. In summary, the considered problem is NP-hard in theory. We provide some discussions about a PPDC in terms of the numbers of constraints and variables. In the worst scenario, the number of the constraints in a PPDC is $(|P_A + 1|)|P_R| + 5|P_A|$. The number of the variables in a PPDC is twice the number of the operation places in a Petri net, i.e., $2|P_A|$.

Compared with most of other transition-based methods that deal with deadlock problems, such as [20] and [21], this study has the following advantages: (1) a reachability graph is not required, which avoids the state explosion problem. Although we must solve ILPPs, the number of the variables grows linearly with the size of a Petri net, which is rather efficient and applicable for large-scale systems. Moreover, the number of the constraints grows polynomially (with a degree of 2) with the number of the places of the given Petri net. (2) All of the transition-based methods can guarantee that all original reachable markings are retained in the controlled system. The proposed approach not only deals with deadlocks but also livelocks. As a result, the controlled net system is live by adjoining the designed DTIs. Compared with our previous work [25], the proposed method is applicable to more general Petri nets.

V. Experimental Results

In this section, two wildly investigated FMS examples are presented to demonstrate the proposed policy. First, the Petri net depicted in Fig. 8(a) is considered, where $P^0 = \{p_8, p_{12}, p_{20}\}$, $P_R = \{p_{18}, p_{19}, p_{21} - p_{25}\}$, and $P_A = \{p_1 - p_7, p_9 - p_{11}, p_{13} - p_{17}\}$. It has 9378 reachable markings with both deadlock problems and livelock problems.

There are 11 resource-operation loops in the resource requirement graph shown in Fig. 8(b), the details of which are shown in Table III. By Algorithm 1, we identify 75 pre-partial deadlocks. Following Algorithm 2, we derive the minimal universal set as $U = \{p_2, p_3, p_6, p_7, p_{10}, p_{14}, p_{15}\}$, which leads to seven DTIs. Table IV outlines the details of the DTIs. Due to the space considerations, the details of the 75 pre-partial deadlocks are not provided. As a result, the controlled net system (after adding the seven DTIs) is live with the 9378 original reachable markings.

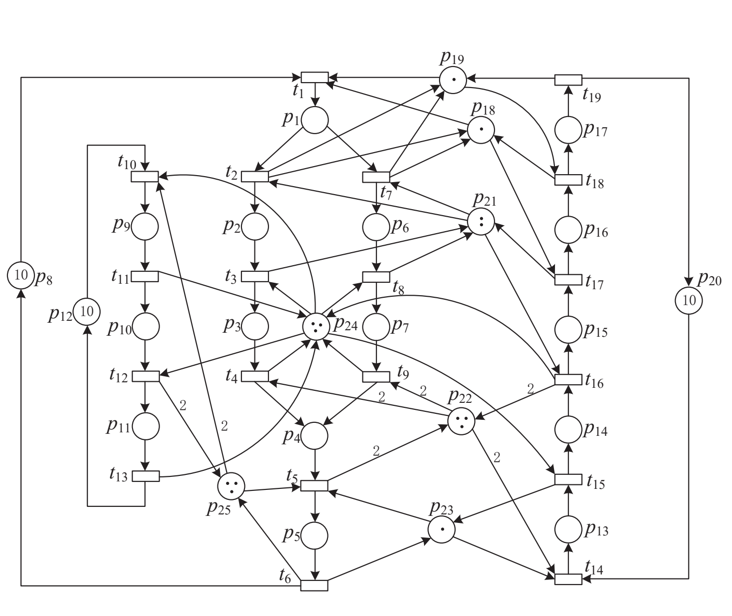
TABLE III: Resource-operation loops in Fig. 8(b)

β_i
$\beta_1 = p_{19} \to (p_1) \to p_{21} \to (p_{15}) \to p_{18} \to (p_{16}) \to p_{19}$
$\beta_2 = p_{18} \to (p_1) \to p_{21} \to (p_{15}) \to p_{18}$
$\beta_3 = p_{21} \to (p_2) \to p_{24} \to (p_{14}) \to p_{21}$
$\beta_4 = p_{21} \to (p_6) \to p_{24} \to (p_{14}) \to p_{21}$
$\beta_5 = p_{24} \to (p_3) \to p_{22} \to (p_{13}) \to p_{24}$
$\beta_6 = p_{24} \to (p_7) \to p_{22} \to (p_{13}) \to p_{24}$
$\beta_7 = p_{24} \to (p_3) \to p_{22} \to (p_4) \to p_{23} \to (p_{13}) \to p_{24}$
$\beta_8 = p_{24} \to (p_7) \to p_{22} \to (p_4) \to p_{23} \to (p_{13}) \to p_{24}$
$\beta_9 = p_{24} \to (p_3) \to p_{22} \to (p_4) \to p_{25} \to (p_9, p_{10}) \to p_{24}$
$\beta_{10} = p_{24} \to (p_7) \to p_{22} \to (p_4) \to p_{25} \to (p_9, p_{10}) \to p_{24}$
$\beta_{11} = p_{24} \to (p_9, p_{10}) \to p_{24}$

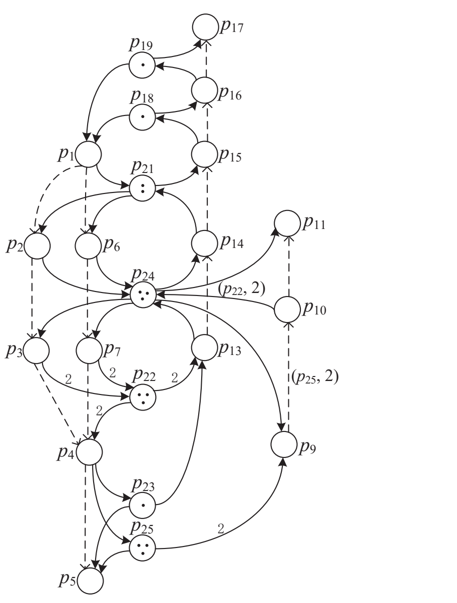
TABLE IV: Details of the seven DTIs

$(\mathcal{M}_{di}, t_{ri})$	$^\bullet t_{ri}$	$t_{ri}{}^\bullet$
$(\mathcal{M}_{d1}, t_{r1})$	p_2	p_8, p_{21}
$(\mathcal{M}_{d2}, t_{r2})$	p_3	p_8, p_{24}
$(\mathcal{M}_{d3}, t_{r3})$	p_6	p_8, p_{21}
$(\mathcal{M}_{d4}, t_{r4})$	p_7	p_8, p_{24}
$(\mathcal{M}_{d5}, t_{r5})$	p_{10}	$p_{12}, 2p_{25}$
$(\mathcal{M}_{d6}, t_{r6})$	p_{14}	$p_{20}, 2p_{22}, p_{24}$
$(\mathcal{M}_{d7}, t_{r7})$	p_{15}	p_{20}, p_{21}

Next, an additional example is given to illustrate the proposed method. The Petri net, shown in Fig. 9(a), has the following place partitions: $P^0 = \{p_1, p_4, p_9\}$, $P_R = \{p_7, p_8\}$,



(a)



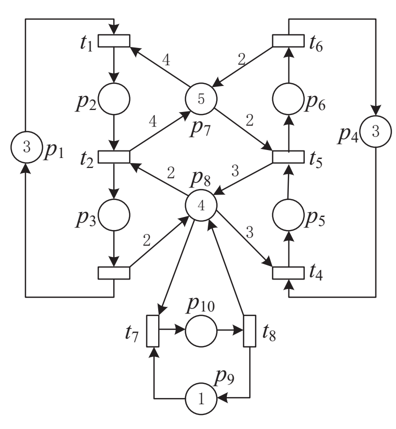
(b)

Fig. 8: (a) Petri net system (N_2, M_{02}), (b) resource requirement graph G_2 of N_2.

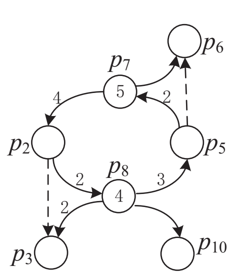
TABLE V: Details of the DTI

$(\mathcal{M}_{di}, t_{ri})$	${}^\bullet t_{ri}$	$t_{ri}{}^\bullet$	$\mathcal{M}_{di}$
$(\mathcal{M}_{d1}, t_{r1})$	p_2	$p_1, 4p_7$	$\{p_2 + p_5\}$

and $P_A = \{p_2, p_3, p_5, p_6, p_{10}\}$. The original net system (N_3, M_{03}) has 24 reachable markings with no deadlock markings. However, the net system is not live, which implies that (N_3, M_{03}) only has livelock problems. According to the resource requirement graph depicted in Fig. 9(b), there is only one resource-operation loop: $\beta_1 = p_7 \to (p_2) \to p_8 \to (p_5) \to p_7$. By Algorithm 1, we have only one pre-partial deadlock, which is $p_2 + p_5$. By Algorithm 2, one DTI is obtained, the details of which are shown in Table V. As a result, the controlled net system (after adding the DTI) is live with 24 reachable markings.



(a)



(b)

Fig. 9: (a) Petri net system (N_3, M_{03}) and (b) resource requirement graph G_3 of N_3.

VI. Conclusion

This paper develops a deadlock detection and recovery policy for FMSs. A resource requirement graph associated with a Petri net is constructed and is used to capture the competition over resources among various processes. By analyzing a resource requirement graph, an algorithm is provided to compute pre-partial deadlocks, described as a set of linear inequalities. Then, a set of recovery transitions that are only enabled at pre-partial deadlock markings is designed by Algorithm 2. This paper tackles both deadlock problems and livelock problems. The controlled net system is live and preserves all reachable markings, without requiring full or partial enumeration of reachable states. Compared with our previous work [25], the proposed deadlock detection and recovery policy is applicable to more general Petri nets. Unlike other related work that transforms deadlocks into legal markings, we only try to release resources held by considered operation places. Our future work will concentrate on developing an on-line deadlock avoidance methodology by exploiting resource requirement graphs.

References

[1] Y. F. Chen, Z. W. Li, and M. C. Zhou, “Optimal supervisory control of flexible manufacturing systems by Petri nets: a set classification approach,” *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 549–563, 2014.

[2] E. G. Coffman, M. J. Elphick, and A. Shoshani, “System deadlocks,” *ACM Computing Surveys*, vol. 3, no. 2, pp. 67–78, 1971.

[3] J. Ezpeleta, J. M. Colom, and J. Martinez, “A Petri net based deadlock prevention policy for flexible manufacturing systems,” *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, 1995.

[4] Z. W. Li and M. Zhao, “On controllability of dependent siphons for deadlock prevention in generalized Petri nets,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 2, pp. 369–384, 2008.

[5] M. V. Iordache, J. Moody, and P. J. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 59–68, 2002.

[6] I. B. Abdallah and H. A. Elmaraghy, "Deadlock prevention and avoidance in FMSs: a Petri net based approach," *International Journal of Advanced Manufacturing Technology*, vol. 14, no. 10, pp. 704–715, 1998.

[7] J. Ezpeleta and L. Recalde, "A deadlock avoidance approach for nonsequential resource allocation systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 1, pp. 93–101, 2004.

[8] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 713–723, 1990.

[9] K. Xing, M. C. Zhou, H. Liu, and F. Tian, "Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 1, pp. 188–199, 2009.

[10] V. Gligor and S. Shattuck, "On deadlock detection in distributed systems," *IEEE Transactions on Software Engineering*, vol. 7, no. 3, pp. 320–336, 1980.

[11] T. K. Kumaran, W. Chang, H. Cho, and A. Wysk, "A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems," *International Journal of Production Research*, vol. 32, no. 10, pp. 2361–2379, 1994.

[12] Z. Ding, M. C. Zhou, and S. Wang, "Ordinary differential equation-based deadlock detection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 10, pp. 1435–1454, 2014.

[13] Z. W. Li and M. C. Zhou, "Clarifications on the definitions of elementary siphons in Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 6, pp. 1227–1229, 2006.

[14] Z. W. Li and M. C. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 4, pp. 313–325, 2006.

[15] D. Y. Chao "Direct minimal empty siphon computation using MIP," *The International Journal of Advanced Manufacturing Technology*, vol. 45, pp. 397–405, 2009.

[16] D. Y. Chao, "Improved controllability test for dependent siphons in S^3PR based on elementary siphons," *Asian Journal of Control*, vol. 12, no. 3, pp. 377–391, 2010.

[17] A. Ghaffari, N. Rezg, and X. L. Xie, "Design of a live and maximally permissive Petri net controller using the theory of regions," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 137–141, 2003.

[18] M. Uzam and M. C. Zhou, "Iterative synthesis of Petri net based deadlock prevention policy for flexible manufacturing systems," in *Proc. IEEE International Conference on Systems, Man and Cybernetics, Hague, The Netherlands*, vol. 5, pp. 4260–4265, 2004.

[19] M. Uzam and M. C. Zhou, "An iterative synthesis approach to Petri net-based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 3, pp. 362–371, 2007.

[20] Y. S. Huang, Y. L. Pan, and P. J. Su, "Transition-based deadlock detection and recovery policy for FMSs using graph technique," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 1, pp. 1–13, 2013.

[21] Y. F. Chen, Z. W. Li, A. Al-Ahmari, N. Wu, and T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with Petri nets," *Information Sciences*, vol. 381, pp. 290–303, 2017.

[22] Y. Y. Dong, Y. F. Chen, S. Li, M. A. El-Meligy, and M. Sharaf, "An efficient deadlock recovery policy for flexible manufacturing systems modeled with Petri nets," *IEEE Access*, vol. 7, pp. 11785–11795, 2019.

[23] M. Bashir, D. Liu, M. Uzam, N. Q. Wu, A. Al-Ahmari, and Z. W. Li, "Optimal enforcement of liveness to flexible manufacturing systems modeled with Petri nets via transition-based controllers," *Advances in Mechanical Engineering*, vol. 10, no. 1, pp. 1687814017750707, 2018.

[24] Y. L. Pan, "One computational innovation transition-based recovery policy for flexible manufacturing systems using Petri nets," *Applied Sciences*, vol. 10, no. 7, pp. 2076–3417, 2020.

[25] Y. Lu, Y. F. Chen, Z. W. Li, and N. Q. Wu, "An efficient method of deadlock detection and recovery for flexible manufacturing systems by resource flow graphs," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1707–1718, 2022.

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

[27] Y. F. Chen, Z. W. Li, K. Barkaoui, and A. Giua, "On the enforcement of a class of nonlinear constraints on Petri nets," *Automatica*, vol. 55, no. 5, pp. 116–124, 2015.

[28] F. S. Hsieh, "Fault-tolerant deadlock avoidance algorithm for assembly processes," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 1, pp. 65–79, 2004.

[29] N. Q. Wu, M. C. Zhou, and Z. W. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 56–69, 2008.

[30] T. K. Kumaran, W. Chang, H. Cho, and A. Wysk, "A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems," *The International Journal of Production Research*, vol. 32, no. 10, pp. 2361–2379, 1994.

[31] Y. Lu, Y. F. Chen, L. Yin, and Z. W. Li, "Optimal transition-based supervisors design for flexible manufacturing systems", *2022 IEEE International Conference on Networking, Sensing and Control (ICNSC), Shanghai, China*, pp. 1–6, 2022.

[32] D. J. Sun, Y. F. Chen, M. A. El-Meligy, M. A. F. Sharaf, N. Q. Wu, and Z. W. 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.

[33] Y. F. Chen, A. Al-Ahmari, C. T. Hon, and N. Q. Wu, "Equivalent transformation of nonlinear constraints to linear constraints in Petri nets," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[34] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, 1996.

[35] Z. W. Li and M. C. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 1, pp. 38–51, 2004.

[36] K. Y. Xing, M. C. Zhou, F. Wang, H. X. Liu, and F. Tian, "Resource-transition circuits and siphons for deadlock control of automated manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 1, pp. 74–84, 2011.

[37] T. C. Row and Y. L. Pan, "Maximally permissive deadlock prevention policies for flexible manufacturing systems using control transition," *Advances in Mechanical Engineering*, vol. 10, no. 7, pp. 1687814018787406, 2018.

[38] T. C. Row, W. M. Syu, Y. L. Pan, and C. C. Wang, " One novel and optimal deadlock recovery policy for flexible manufacturing systems using iterative control transitions strategy," *Mathematical Problems in Engineering*, vol. 2019, pp. 12, 2019.