Robust Deadlock Avoidance Policy for Automated Manufacturing System With Multiple Unreliable Resources

Jianchao Luo, Zhiqiang Liu, Shuogang Wang, and Keyi Xing

Abstract — **This work studies the robust deadlock control of automated manufacturing systems with multiple unreliable resources. Our goal is to ensure the continuous production of the jobs that only require reliable resources. To reach this goal, we propose a new modified Banker's algorithm (MBA) to ensure that all resources required by these jobs can be freed. Moreover, a Petri net based deadlock avoidance policy (DAP) is introduced to ensure that all jobs remaining in the system after executing the new MBA can complete their processing smoothly when their required unreliable resources are operational. The new MBA together with the DAP forms a new DAP that is robust to the failures of unreliable resources. Owing to the high permissiveness of the new MBA and the optimality of the DAP, it is tested to be more permissive than state-of-the-art control policies.**

Index Terms—**Automated manufacturing system (AMS), deadlock avoidamce policy (DAP), modified Banker's algorithm (MBA), Petri net.**

NOMENCLATURE

Z	$\{0, 1, 2, \ldots\}$.
Z^+	$\{1, 2, 3, \ldots\}$.
Z_k	$\{1, 2, \ldots, k\}$ where $k \in Z^+$.
n	The number of job types.
m	The number of resources.
$\mathbb{S}$	$\{R, J, Q, \sum, \xi, \delta\}$, automata model of the whole system.
R	$R^U \cup R^R$, set of resources.
R^U	Set of unreliable resources.
R^R	Set of reliable resources.
J	$\{J_1, J_2, ..., J_n\}$, set of job (or part) types.
J_{ik}	The kth operation stage of $J_i \in J$.
$\rho(J_{ik})$;	The resource required by J_{ik}.
$\Pi(r)$	$\{J_{ik}: \rho(J_{ik}) = r\}$, set of operation stages that require resource r.
Q	Set of all system states.
q_0	Initial state.
$Q_R(q_0)$	Set of states reachable from q_0.
$Q_R(q_0, \Delta)$	Set of states reachable from q_0 under policy Δ.
Σ	$\Sigma_c \cup \Sigma_u$, the set of all system events.
Σ_c	Set of controllable events.
Σ_u	Set of uncontrollable events.
ξ	A mapping that associates $q \in Q$ to a set of events enabled at q, i.e., $\xi(q)$.
δ	A mapping that associates $q \in Q$ and an event $\pi \in \xi(q)$ to a new state, i.e., $\delta(q, \pi)$.
N	$(P \cup P_R, T, F)$, PN model of the whole system.
P	$\cup_{i \in Z_n} P_i$, set of operation places.
P_i	$\{p_{i1}, \ldots, p_{i\lvert J_i \rvert}\}$, set of operation places of $J_i \in J$.
T	$\cup_{i \in Z_n} T_i$, set of transitions.
T_i	$\{t_{i1}, \ldots, t_{i(\lvert J_i \rvert+1)}\}$, set of transitions corresponding to J_i.
M_0	Initial marking of N.
$\mathbb{R}(N, M_0)$	Set of all reachable markings of N from M_0.
χ	A mapping that associates a state $q \in Q_R(q_0)$ to a marking $M \in \mathbb{R}(N, M_0)$.
FD	Failure dependent.
R_F	Set of all FD resources.
$R_F(r_j)$	Set of FD resources on $r_j \in R^U$.
ϑ_{ij}	$\langle \rho(J_{ij}), \ldots, \rho(J_{i\lvert J_i \rvert}) \rangle$ denote the remaining route of J_{ij}.
N_f	$(P_f \cup R_F, T_f, F_f)$, failure-affected subnet of N.
γ	A mapping that associates a marking $M \in \mathbb{R}(N, M_0)$ to a marking M_1 of N_f.
N_{fi}	$(P_{fi} \cup R_F(r_i), T_{fi}, F_{fi})$, subnet corresponding to $r_i \in R^U$.
$\mathcal{R}(q)$	Set of failed resources at q.

Manuscript received October 9, 2019; accepted December 17, 2019. This work was supported in part by the Fundamental Research Funds for the Central Universities (3102017OQD110), the Natural Science Basic Research Plan in Shaanxi Province of China (2019JQ-435), the Project Funded by China Postdoctoral Science Foundation (2019M663818), the National Key Research and Development Program of China (2019YFB1703800), Guangdong Basic and Applied Basic Research Foundation (2019A1515111076), and the National Natural Science Foundation of China (71931007). Recommended by Associate Editor MengChu Zhou. *(Corresponding author: Jianchao Luo.)*

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

J. C. Luo, Z. Q. Liu, and S. G. Wang are with the School of Software, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: luojianchao@nwpu.edu.cn; zqliu@nwpu.edu.cn; 18792913220@163.com).

K. Y. Xing is with the State Key Laboratory for Manufacturing Systems Engineering, and Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: kyxing@sei.xjtu.edu.cn).

Digital Object Identifier 10.1109/JAS.2020.1003096

I. INTRODUCTION

WITH the rapid development of information technology, a growing number of traditional manufacturing systems are reformed into automated manufacturing systems (AMSs) to improve the economy profit and enhance the competitiveness. As a consequence, the degree of resource sharing in AMSs becomes increasingly high. The shared resources together with the interacting parts may cause deadlocks under which the whole system or a part of it stalls indefinitely.

Therefore, many researchers focus on solving such deadlocks, and propose different kinds of control policies [1]–[17]. Most of them are only applicable for AMSs without unreliable resources.

Although a great deal of techniques have been applied to extend the useful life of resources in AMSs, unreliable resources are still inevitable. The failures of unreliable resources will block the production of the jobs that require them. Since the resources occupied by the blocked jobs are required by other jobs, such an occupation may cause the whole AMS stalled. Thus, developing efficient control policies to deal with such blockages in AMSs is necessary.

More and more researchers realize the importance and propose different kinds of robust control policies [18]–[36]. Liu *et al.* [27] concentrate on the robust control of AMS with unreliable resources. They model the system by the Petri net (PN) and develop a PN-based controller to ensure its liveness. Wu *et al.* [33] investigate the system studied in [27]. They construct a constraint set for each minimal siphon and add a controller to restrict its activities. Differing from the controller in [27], their controller can ensure the continuous production of all jobs even if some unreliable resource breaks down.

All resources in the AMSs in [21], [22], [25]–[27], [33] are machines or robots, whereas those in [18]–[20], [23], [24], [28]–[32], [34]–[36] are workstations with a machine for processing jobs and a buffer space for storing jobs. Lawley [23] pioneers the study on the control of such AMS with an unreliable resource. By combining a set of resource order constraints with a set of neighborhood constraints, he proposes a robust control policy. In order to make a controlled AMS more permissive, five different methods have been proposed in [24], [28], [29], [32], and [36], respectively. Chew *et al.* [18] pioneer the study on the control of AMS with multiple unreliable resources. They develop a robust DAP that is made up of an MBA and several neighborhood constraints. Yue *et al.* [34] improve the permissiveness of the MBA in [18] by moving the jobs that finish their operations on unreliable resources out of the system to release buffers. Chew *et al.* [19] and Yue *et al.* [35] present two solutions to relax the restriction on AMSs in [18] that a job requires no more than one unreliable resource.

An unreliable resource may fail when it is working (processing jobs) because its bearing may be worn out. Also, it may fail when it is idle (not processing jobs) because its electronic components may be affected with damp. We study the robust control problem of AMSs with multiple unreliable resources that may fail when they are working or idle. The studied AMSs are different with those in [18], [19], [30], [34], [35], where unreliable resources can fail only when they are working. The studied AMSs are modeled by automata models and PN models. Based on the developed models, a new version of MBA is developed and the PN-based DAP in [16] is introduced. They together ensure that AMS can always produce those jobs that only require operational resources if no additional unreliable resources fail. Compared with the controllers in [18], [20], [23], [24], [27], [28], [31], [32], [34]–[36], ours owns higher permissiveness. Moreover, the system controlled under it is also robust to the failure of unreliable resources that may fail when they are idle.

Main contributions are summarized as below.

1) A new MBA is proposed, which is proved to be more permissive than the improved version in [34].

2) The new proposed MBA can be integrated with any DAP for system of simple sequential process with resources (S^3PR [3]) in the same way as with the DAP in [16], which eventually leads to improved performance of our control policy if a better DAP for S^3PR is proposed.

3) We study the robust deadlock control of AMSs where resources are workstations and unreliable resources may fail when they are working or idle for the first time.

4) The proposed robust DAP exhibits higher permissiveness than existing ones for systems with multiple unreliable resources or a single one.

The rest paper is organized as follows. Section II models the studied AMSs. A robust DAP is developed in Section III. Section IV illustrates its effectiveness. The whole work is concluded in Section V.

II. Modeling of AMSs

This section describes the considered AMSs. Thereafter, their automata and PN models are established. The following two reasons encourage us to develop both automata and PN models. Firstly, almost all existing control policies for the studied systems are established based on automata models. Theoretical comparison with existing policies is necessary to illustrate the effectiveness of our policy. Such a comparison can only be conducted under automata models. Secondly, we will introduce a PN-based optimal DAP that is proposed for AMSs without unreliable resources to the studied AMSs. The introduced policy is not understandable under the automata model.

A. Automata Model

A resource in the studied AMS has a machine and a buffer space, which are used to process and store jobs, respectively. A buffer space may contain several buffers. Each buffer can hold a job at a time. All buffer spaces are always reliable, while several machines are unreliable. A resource is unreliable if it contains an unreliable machine. Unreliable resources may fail when they are working or idle. Suppose that their failures never destroy the jobs being processed. Each job requires no more than one unreliable resource. Note that the studied AMSs are more general than those in [18] and [34], where unreliable resources can fail only when they are working.

The system is described as a six-tuple $\mathbb{S} = \{R, J, Q, \Sigma, \xi, \delta\}$, where $R = \{r_1, r_2, \ldots, r_m\} = R^U \cup R^R$ with R^U (R^R) being the set of unreliable (reliable) resources, $R^U \neq \emptyset$, $R^R \neq \emptyset$, and $R^U \cap R^R = \emptyset$. Let $C(r)$ denote the number of buffers of resource r.

J represents the set of job (or part) types. $J_i = \langle J_{i1}, J_{i2}, \ldots, J_{i|J_i|}\rangle \in J$ is a sequence of operation stages with J_{ik} being its kth operation stage. Let $\rho(J_{ik})$ denote the resource required by J_{ik}. All job instances of J_{ik} dwell in the buffers of $\rho(J_{ik})$. Let $\Pi(r) = \{J_{ik}: \rho(J_{ik}) = r\}$ denote the set of operation stages that require resource r.

Q is the set of all system states. $q = \langle \lambda_i, x_{jk}, y_{jk}: i \in Z_{|R^U|}, j \in Z_{|J|}, k \in Z_{|J_j|}\rangle \in Q$ is a state with $\lambda_i \in \{1, 0\}$ being the status of $r_i \in R^U$ (1 if operational, 0 if failed), and x_{jk} (y_{jk}) $\in Z$

being the number of finished (unfinished) job instances of J_{jk}. Specially, $q_0 \in Q$ is the initial state with $\lambda_i = 1$ for $i \in Z_{|R^U|}$ and $x_{jk} = y_{jk} = 0$ for $j \in Z_{|J|}$, $k \in Z_{|J_j|}$.

$\Sigma = \Sigma_c \cup \Sigma_u$ represents the set of all system events with Σ_c (Σ_u) being the set of controllable (uncontrollable) events. $\Sigma_c = \{\alpha_{jk}: j \in Z_{|J|}, k \in Z_{|J_j|+1}\}$, where α_{jk} ($k \le |J_j|$) represents the assignment of $\rho(J_{jk})$ to a job instance of J_{jk}, and $\alpha_{j(|J_j|+1)}$ represents the movement of a finished job of J_j out of $\mathbb{S}$. $\Sigma_u = \Sigma_{u1} \cup \Sigma_{u2}$, where $\Sigma_{u1} = \{\beta_{jk}: j \in Z_{|J|}, k \in Z_{|J_j|}\}$ with β_{jk} representing the completion of service for a job instance of J_{jk} and $\Sigma_{u1} = \{\kappa_i, \eta_i: i \in Z_{|R^U|}\}$ with κ_i (η_i) representing the failure (repair) of $r_i \in R^U$.

ξ: $Q \rightarrow 2^\Sigma$ is a mapping that associates $q \in Q$ to a set of events enabled at q, i.e., $\xi(q)$, which is defined below. a) $\forall r_i \in R$, $\forall J_{t1} \in \Pi(r_i)$, if $C(r_i) - \Sigma_{J_{kv} \in \Pi(r_i)}(x_{kv} + y_{kv}) > 0$, then $\alpha_{t1} \in \xi(q)$; b) $\forall r_i \in R^R$, $\forall J_{tl} \in \Pi(r_i)$, if $y_{tl} > 0$, then $\beta_{tl} \in \xi(q)$; c) $\forall r_i \in R^U$ and $\lambda_i = 1$, $\forall J_{tl} \in \Pi(r_i)$, if $y_{tl} > 0$, then $\beta_{tl} \in \xi(q)$; d) $\forall r \in R$, $\forall J_{tl} \in \Pi(r)$ with $l > 1$ and $l \in Z_{|J_t|}$, if $x_{t(l-1)} > 0$ and $C(r) - \Sigma_{J_{kv} \in \Pi(r)}(x_{kv} + y_{kv}) > 0$, then $\alpha_{tl} \in \xi(q)$; e) if $x_{t|J_t|} > 0$, then $\alpha_{t(|J_t|+1)} \in \xi(q)$; and f) $\forall r_i \in R^U$ and $\lambda_i = 1$, then $\kappa_i \in \xi(q)$; g) $\forall r_i \in R^U$ and $\lambda_i = 0$, then $\eta_i \in \xi(q)$.

δ: $Q \times \Sigma \rightarrow Q$ is a mapping that associates $q \in Q$ and an event $\pi \in \xi(q)$ to a new state, i.e., $\delta(q, \pi)$, which is obtained after enabling π at q.

Note that the above automata model and that in [35] only differ in the condition for resource failures. In this work, $r_i \in R^U$ may fail if $\lambda_i = 1$; while in [35], $r_i \in R^U$ may fail if $\lambda_i = 1$ and $\exists J_{jk} \in \Pi(r_i) \ni y_{jk} > 0$.

Let $\sigma = \pi_1\pi_2\cdots\pi_k$ be a string of events of Σ. σ is feasible from $q \in Q$ if $\forall i \in Z_k$, $\exists q_{i+1} = \delta(q_i, \pi_i) \in Q$, where $q_1 = q$. Then, q_{i+1} is called reachable from q. Let $Q_R(q)$ represent the set of states reachable from q, and $Q_R(q, \Delta)$ represent the set of states reachable from q under control policy Δ.

B. PN Model

We omit the preliminaries of PN in our paper. They can be found in [37]–[49]. The PN model of a studied system is defined below.

Definition 1: A studied AMS is modeled by a PN $N = (P \cup P_R, T, F)$ where

a) $P = \cup_{i \in Z_n} P_i$, where $P_i = \{p_{i1}, \ldots, p_{i|J_i|}\}$, $P_i \ne \emptyset$, $P_i \cap P_j = \emptyset$ $\forall i, j \in Z_n$ and $i \ne j$;

b) $P_R = \{r_1, \ldots, r_m\}$, where $r_i \in P_R$ represents the buffer space of resource r_i;

c) $T = \cup_{i \in Z_n} T_i$, where $T_i = \{t_{i1}, \ldots, t_{i(|J_i|+1)}\}$, $T_i \ne \emptyset$, $T_i \cap T_j = \emptyset$ $\forall i, j \in Z_n$ and $i \ne j$;

d) $\forall i \in Z_n$, $P_i \cup T_i$ generates a subnet that is a state machine;

e) $\forall i \in Z_m$, $\exists$a minimal P-Semiflow Y_i such that $||Y_i|| \cap P_R = \{r_i\}$, $Y_i(r_i) = 1$, $||Y_i|| \cap P \ne \emptyset$; and

f) $P = \cup_{i \in Z_m}\{||Y_i|| \backslash \{r_i\}\}$.

A brief description of above mentioned definitions is as following:

i) Definition 1-a) suggests that $p_{ij} \in P_i$ is the operation place of J_{ij}.

ii) Definition 1-c) suggests that $t_{ij} \in T_i$ is a transition which is corresponded to α_{ij}. Operation place p_{ij} requires a resource $\rho(p_{ij}) = \rho(J_{ij})$. Thus, the PN model is corresponded to the automata model. As we stated earlier, the introduced PN-based DAP is proposed for AMSs without unreliable resources. It cannot handle uncontrollable events. Thus, only the controllable events are modeled in the PN model. For more details about the PN models of uncontrollable events, please refer to [35].

iii) Definition 1-e) restricts that all buffer spaces can never be created or destroyed and they must be used by some operations. $\forall i \in Z_m$, $H(r_i) = ||Y_i|| \backslash \{r_i\} = \{p_{jk} \mid \rho(p_{jk}) = r_i\}$ denotes the set of operation places that use r_i.

The initial marking of N, i.e., M_0, represents no job instance in the system. Thus, $\forall p \in P$, $M_0(p) = 0$, and $\forall r_i \in P_R$, $M_0(r_i) = C(r_i)$. All markings of N reachable from M_0 is represented by $\mathbb{R}(N, M_0)$.

Since the PN model is corresponded to the automata model, a state of $Q_R(q_0)$ is corresponded to a marking of $\mathbb{R}(N, M_0)$. Let χ: $Q_R(q_0) \rightarrow \mathbb{R}(N, M_0)$. $\forall q \in Q_R(q_0)$, $M = \chi(q)$ is defined below. $\forall p_{jk} \in P$, $M(p_{jk}) = x_{jk} + y_{jk}$, and $\forall r_i \in P_R$, $M(r_i) = C(r_i) - \Sigma_{p_{jk} \in H(r_i)} M(p_{jk})$.

There is a difference between the above PN model and that in [35], i.e., our PN models all controllable events while only a part of controllable events are modeled in the PN in [35].

Example 1: Consider a system with $R^R = \{r_1, r_2, r_3, r_5, r_6\}$, $R^U = \{r_4, r_7\}$, $C(r_2) = 2$, $C(r_1) = C(r_3) = C(r_4) = C(r_5) = C(r_6) = C(r_7) = 1$, and $J = \{J_1, J_2, J_3, J_4\}$. $|J_1| = 3$, $|J_2| = |J_3| = |J_4| = 4$, $\rho(J_{11}) = \rho(J_{23}) = \rho(J_{31}) = r_1$, $\rho(J_{12}) = \rho(J_{22}) = \rho(J_{43}) = r_2$, $\rho(J_{13}) = \rho(J_{21}) = r_3$, $\rho(J_{24}) = r_4$, $\rho(J_{32}) = \rho(J_{42}) = r_5$, $\rho(J_{33}) = \rho(J_{41}) = r_6$, and $\rho(J_{34}) = \rho(J_{44}) = r_7$. Its PN model is shown in Fig. 1.

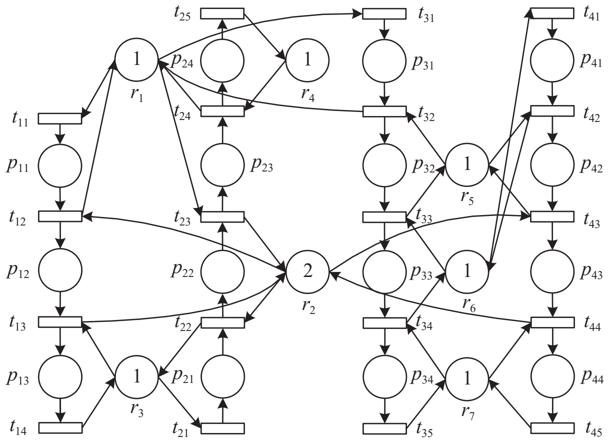


Fig. 1. PN model of the system in Example 1.

Let ${}^{(o)}t$ ($t^{(o)}$) represent the input (output) operation places of $t \in T$, and ${}^{(r)}t$ ($t^{(r)}$) represent its input (output) resource places. $t \in T$ is enabled at $M \in \mathbb{R}(N, M_0)$ if (a) $M({}^{(o)}t) > 0$ (t is process-enabled) and (b) $M({}^{(r)}t) > 0$ (t is resource-enabled). Only enabled transitions can fire.

III. Robust Deadlock Avoidance Policy

Initially, the properties that a robust control policy for AMSs with multiple unreliable resources should satisfy are reviewed. Later on, a DAP is proposed to satisfy them.

A. Properties for a Robust Control Policy

The properties that a robust control policy should have are firstly discussed in [18], where motivating examples for these properties can be found. For economy of space, we recall them briefly.

If all jobs not requiring failed resources are possible to be completed from a state, then the state is called as a feasible initial (FI) state. A robust control policy should guarantee that a state of a system can always be treated as an FI state. Formally, the properties presented in [18] are recalled as follows.

Definition 2 [18]: A control policy Δ is robust to the failures of R^U if a) it guarantees the continuous production of jobs not requiring failed resources if no additional failures or repairs occur; b) it only admits states that are FI if some additional failure occurs; and c) it only admits states that are FI if some additional repair occurs.

By Definition 2, we know that a robust control policy should guarantee that the system operates normally when no resource fails, which can be found in Definition 2-a). Also, it should guarantee that the system can normally process those jobs not requiring failed resources when a part of the unreliable resources fail, which can be found in Definition 2-b) and 2-c).

The aim of this work is to propose a robust DAP that satisfies properties in Definition 2. Note that such an aim is the same as that in [18] and [34]. Since the studied AMSs in this work are more general than those in [18] and [34], our work is more challenging than those in [18] and [34]. The proposed robust DAP contains two parts: a new MBA and a PN-based DAP, which are given as follows.

B. A New Modified Banker's Algorithm

Before proposing the new MBA ($\mathrm{MBA_L}$), we review some definitions about the classification of resources and operation stages as follows.

Definition 3 [18]: A resource $r_i \in R$ is failure dependent (FD) on unreliable resource $r_j \in R^U$ if, $\forall J_{st} \in \Pi(r_i)$, $\exists J_{s(t+c)} \in \Pi(r_j)$, where $c \in Z^+$. Let $R_F(r_j)$ be the set of FD resources on r_j and $R_F = \cup_{r_j \in R^U} R_F(r_j)$ be the set of all FD resources.

Definition 4 [34]: Let $\mathbb{J}$ be the set of all operation stages. Given $J_{ij} \in \mathbb{J}$, let $\vartheta_{ij} = \langle \rho(J_{ij}), \ldots, \rho(J_{i|J_i|})\rangle$ denote its remaining route. Let $\mathbb{J}^1 = \{J_{ij}: \vartheta_{ij} \cap R_F = \emptyset\}$, $\mathbb{J}^2 = \{J_{ij}: \rho(J_{ij}) \in R^U$ and $\vartheta_{i(j+1)} \cap R_F = \emptyset\}$, $\mathbb{J}^3 = \{J_{ij}: \vartheta_{ij} \cap R_F \neq \emptyset$ and $\rho(J_{ij}) \notin R_F\}$, and $\mathbb{J}^4 = \mathbb{J}\backslash(\mathbb{J}^1 \cup \mathbb{J}^2 \cup \mathbb{J}^3)$. Let $q \in Q$ be a state, $\mathbb{J}^+(q) = \{J_{ij} \in \mathbb{J}: x_{ij} + y_{ij} > 0\}$, $\mathbb{J}^1(q) = \mathbb{J}^+(q) \cap \mathbb{J}^1$, $\mathbb{J}^2(q) = \mathbb{J}^+(q) \cap \mathbb{J}^2$, $\mathbb{J}^3(q) = \mathbb{J}^+(q) \cap \mathbb{J}^3$, and $\mathbb{J}^4(q) = \mathbb{J}^+(q) \cap \mathbb{J}^4$.

By Definition 4, $\mathbb{J}^1$ is the set of operation stages whose jobs do not require any FD resources in their remaining routes. $\mathbb{J}^2$ is the set of operation stages whose jobs occupy unreliable resources but those jobs in their next stages do not require any FD resources in their remaining routes. $\mathbb{J}^3$ is the set of operation stages whose jobs require FD resources in their remaining routes but do not occupy FD resources currently.

Example 2: Consider the system in Fig. 1, $R^U = \{r_4, r_7\}$, $R_F(r_4) = \{r_4\}$, $R_F(r_7) = \{r_5, r_6, r_7\}$, and $R_F = \{r_4, r_5, r_6, r_7\}$. $\mathbb{J} = \{J_{11}–J_{13}, J_{21}–J_{24}, J_{31}–J_{34}, J_{41}–J_{44}\}$, $\mathbb{J}^1 = \{J_{11}–J_{13}\}$, $\mathbb{J}^2 = \{J_{24}, J_{34}, J_{44}\}$, and $\mathbb{J}^3 = \{J_{21}–J_{23}, J_{31}, J_{43}\}$, and $\mathbb{J}^4 = \{J_{32}, J_{33}, J_{41}, J_{42}\}$.

Given $q \in Q$, $\mathrm{MBA_L}$ tries to move 1) all jobs at stage $J_{ij} \in \mathbb{J}^1(q)$ out of $\mathbb{S}$; 2) all jobs at stage $J_{ij} \in \mathbb{J}^2(q)$ that have finished their current operations on $\rho(J_{ij})$ out of $\mathbb{S}$; and 3) all jobs at stage $J_{ij} \in \mathbb{J}^3(q)$ into the first-met FD resource in ϑ_{ij}. Let $\mathbb{J}(q) = \mathbb{J}^1(q) \cup \mathbb{J}^2(q) \cup \mathbb{J}^3(q)$ represent the set of operation stages considered by $\mathrm{MBA_L}$ at q.

A different version of MBA that considers the same set of operation stages has been proposed in [34]. It is denoted as $\mathrm{MBA_Y}$. Given $q \in Q$, $\mathrm{MBA_Y}$ tries to find an ordering of $\mathbb{J}(q)$ such that the jobs at the according operation stages can be moved to proper positions sequentially. If such an ordering is not found, it rejects q.

Note that even if such an ordering does not exist, there may still be a string of events by which all jobs can be moved to proper positions. To see this, let us reconsider Example 1. Consider a state $q = 1 + 1 + x_{11} + x_{21} \in Q$. $\mathbb{J}(q) = \{J_{11}, J_{21}\}$, $\mathbb{J}^1(q) = \{J_{11}\}$, and $\mathbb{J}^3(q) = \{J_{21}\}$. The first-met FD resource in ϑ_{21} is r_4. Thus, $\mathrm{MBA_Y}$ tries to move the job at stage J_{11} out of the system and that at stage J_{21} into r_4. A job at stage J_{11} occupies the only one buffer of r_1 and needs to be processed on r_3, while a job at stage J_{21} occupies the only one buffer of r_3 and needs to be processed on r_1. None of them can be moved to their proper positions directly. Thus, the $\mathrm{MBA_Y}$ rejects q. In fact, after moving the job at stage J_{11} to r_2 from q, the system reaches $q_1 = 1 + 1 + y_{12} + x_{21}$. There is an ordering of $\mathbb{J}(q_1)$, i.e., $J_{21}J_{12}$ such that all jobs can be moved to their proper positions sequentially. Motivated by such an example, we modify the $\mathrm{MBA_Y}$ further and propose a new version of MBA, i.e., $\mathrm{MBA_L}$. $\mathrm{MBA_L}$ can admit more states than $\mathrm{MBA_Y}$ does, to be shown later.

Given $q \in Q$, let $\zeta:\{1, \ldots, |\mathbb{J}(q)|\} \rightarrow \mathbb{J}(q)$ be an indexing function on $\mathbb{J}(q)$ with $\zeta(i)$ ($i \in Z_{|\mathbb{J}(q)|}$) being the ith operation stage of $\mathbb{J}(q)$. In $\mathrm{MBA_L}$, Λ_a is an array with $\Lambda_a[j]$ recording the number of idle buffers of $r_j \in R$. Λ_r is a two-dimensional array with $\Lambda_r[i][j] = 1$ if a job at $\zeta(i)$ requires $r_j \in R$ to be moved to its proper position, and otherwise $\Lambda_r[i][j] = 0$. Λ_o is another two-dimensional array with $\Lambda_o[i][j] = x_{st} + y_{st}$ if $\zeta(i) = J_{st}$ and $\rho(J_{st}) = r_j$, and otherwise $\Lambda_o[i][j] = 0$.

$\mathrm{MBA_L}$ contains two main parts: Initialization and $\mathrm{MBA_1}$. Initialization initializes Λ_a, Λ_r, and Λ_o for an input state q. $\mathrm{MBA_1}$ is equivalent to the main body of $\mathrm{MBA_Y}$. Given a state q, $\mathrm{MBA_L}$ invokes Initialization to initializes Λ_a, Λ_r, and Λ_o for q. Then, $\mathrm{MBA_L}$ invokes $\mathrm{MBA_1}$ to judge if q is admitted by $\mathrm{MBA_1}$. If it is admitted, $\mathrm{MBA_L}$ accepts it too. Such a process corresponds to Lines 1−4 in $\mathrm{MBA_L}$. Otherwise, $\mathrm{MBA_L}$ tries to find a job remaining in the system that can be moved a step forward. If such a job is found, $\mathrm{MBA_L}$ invokes $\mathrm{MBA_1}$ to judge if the state obtained after moving such a job is admitted by $\mathrm{MBA_1}$. If it is admitted, $\mathrm{MBA_L}$ accepts it too. Otherwise, $\mathrm{MBA_L}$ tries to move such a job two steps forward and invokes $\mathrm{MBA_1}$ to judge the state obtained after such a movement. If it is admitted, $\mathrm{MBA_L}$ accepts it too. This process repeats until $\mathrm{MBA_L}$ accepts the resulting state or such a job cannot be moved forward more. Such a process corresponds to Lines 8−20 in $\mathrm{MBA_L}$. If such a job cannot be moved forward more, $\mathrm{MBA_L}$ tries to find another job remaining in the system that can be moved a step forward. Then, the process corresponding to Lines 8−20 repeats. Such a process repeats until all jobs remaining in the system have been tried. This process corresponds to the for-loop in Lines 6-21. Finally, MBA_L rejects q if all jobs have been tried and no resulting state is accepted by MBA_1. Formally, Algorithm I (MBA_1), Algorithm II (Initialization), and Algorithm III (MBA_L) are as follows.

Algorithm I. Modified Banker's Algorithm MBA_1

```
Input: a state $q \in Q$, $\Lambda_a$, $\Lambda_r$, $\Lambda_o$;
Output: $(0, q_e)$ or $(1, q_e)$; /*$\{0, 1\}\times Q$*/;
1: $q_e = q$; $I = Z_{|\mathbb{J}(q)|}$; /* initialization */
2: while ($I \neq \emptyset$)
3:   find $i \in I \ni \forall j \in Z_m, \Lambda_r[i][j] \leq \Lambda_a[j]$;
4:   if no such $i$ exists, return $(0, q_e)$;
5:   if ($\zeta(i) = J_{st} \in \mathbb{J}^1$)
6:     move all jobs at $J_{st}$ out of $\mathbb{S}$ and reach a new state $q_n$;
7:     $\Lambda_a[j] = \Lambda_a[j] + \Lambda_o[i][j]$, where $\rho(J_{st}) = r_j$;
8:     $I = I\setminus\{i\}$;
9:   else if ($\zeta(i) = J_{st} \in \mathbb{J}^2$)
10:      move those jobs at stage $J_{st}$ which have finished their operations on $\rho(J_{st})$ out of $\mathbb{S}$ and reach a new state $q_n$;
11:    $\Lambda_a[j] = \Lambda_a[j] + x_{st}$, where $\rho(J_{st}) = r_j$;
12:    $I = I\setminus\{i\}$; }
13:  else /*$\zeta(i) = J_{st} \in \mathbb{J}^3$*/
14:    let $r_j$ be the first-met FD resource in $\vartheta_{st}$ and $\rho(J_{st}) = r_k$;
15:    if ($r_j \in R^U$)
16:      move a job at $J_{st}$ to $r_j$, and reach a new state $q_n$;
17:      $\Lambda_a[k] = \Lambda_a[k] + 1$; $\Lambda_a[j] = \Lambda_a[j] - 1$;
18:      $\Lambda_o[i][k] = \Lambda_o[i][k] - 1$;
19:      if ($\Lambda_o[i][k] = 0$), $I = I\setminus\{i\}$;
20:    else
21:      move all jobs at $J_{st}$ to $r_j$, and reach a new state $q_n$;
22:      $\Lambda_a[k] = \Lambda_a[k] + \Lambda_o[i][k]$, $\Lambda_a[j] = \Lambda_a[j] - \Lambda_o[i][k]$;
23:      $I = I\setminus\{i\}$;
24:      if ($\Lambda_a[j] < 0$), return $(0, q_e)$;
25:    end if-else
26:  end if-else if-else
27:  $q_e = q_n$;
28: end while
29: return $(1, q_e)$;
```

Lemma 1: Algorithm I is of polynomial complexity.

Proof: Let $s_1 = \Sigma_{i\in Z_n}|J_i|$ and $s_2 = \Sigma_{r\in R^U}C(r)$. There are at most s_1 elements in I and at most s_2 jobs on unreliable resources, thus the while-loop in Algorithm I can be iterated at most $(s_1 + s_2)$ times. In the while-loop, all Lines except for Line 3 can be completed in a constant time. The complexity of Line 3 is $O(s_1 \times m)$. Thus the total complexity of Algorithm I is $O((s_1 + s_2) \times s_1 \times m)$. ■

Algorithm II. Initialization

```
Input: a state $q \in Q$;
Output: $\Lambda_a$, $\Lambda_r$, $\Lambda_o$;
1: $I = Z_{|\mathbb{J}(q)|}$; /* index set of $\mathbb{J}(q)$ */
2: for $i \in I$ and $j \in Z_m$
3:   $\Lambda_r[i][j] = 0$; $\Lambda_o[i][j] = 0$;
4: end for
5: for $i \in I$
6:   if ($\zeta(i) = J_{st}$ and $\rho(J_{st}) = r_j$), $\Lambda_o[i][j] = x_{st} + y_{st}$;
7: end for
8: for $j \in Z_m$
9:   $\Lambda_a[j] = C(r_j) - \Sigma_{i\in I}\Lambda_o[i][j]$;
10: end for
11: for $i \in I$
12:   if ($\zeta(i) = J_{st} \in \{\mathbb{J}^1 \cup \mathbb{J}^3\}$)
13:     for $j \in Z_m$
14:       if ($r_j \in \vartheta_{st}$ ^ $r_j \neq \rho(J_{st})$)
15:         $\Lambda_r[i][j] = 1$;
16:       end for
17:   if ($\zeta(i) = J_{st} \in \mathbb{J}^2$)
18:       for $j \in Z_m$
19:         if ($r_j \in \{\rho(J_{s(t+l)})$: $l \in Z_c$ and $c = \min\{k$: $k > 0$ and $\rho(J_{s(t+k)}) \in R_F\}\}$ ^ $r_j \neq \rho(J_{st})$)
20:         $\Lambda_r[i][j] = 1$;
21:       end for
22: end for
23: return $(\Lambda_a, \Lambda_r, \Lambda_o)$
```

Algorithm III. MBA_L

```
Input: a state $q \in Q$;
Output: $\text{MBA}_\text{L}(q)$; /*$\{0, 1\}\times Q$*/;
1: $q_e = q$; $I = Z_{|\mathbb{J}(q)|}$; /* initialization */
2: $(\Lambda_a, \Lambda_r, \Lambda_o)$ = Algorithm II($q$);
3: (flag, $q_e$) = Algorithm I($q, \Lambda_a, \Lambda_r, \Lambda_o$);
4: if (flag = 1), return $(1, q_e)$;
5: else
6:   for $i \in I$
7:     $q_1 = q_e$;
8:     if ($\zeta(i) = J_{st} \in \{\mathbb{J}^1 \cup \mathbb{J}^3\}$)
9:       let $j_{\max} = |J_s|$;
10:    else /*$\zeta(i) = J_{st} \in \mathbb{J}^2$*/
11:      let $j_{\max} = \min\{k$: $k \in Z^+$ and $\rho(J_{s(t+k)}) \in R_F\}$;
12:    end if-else
13:    for ($j = t + 1$; $j < j_{\max}$; $j$++)
14:      if (it is possible to move a job at $J_{st}$ to $J_{sj}$)
15:          let $\sigma$ be a string of events corresponding to such a movement, and $q_2 = \delta(q_1, \sigma)$;
16:        $(\Gamma_a, \Gamma_r, \Gamma_o)$ = Algorithm II($q_2$);
17:        (flag, $q_{e1}$) = Algorithm I($q_2, \Gamma_a, \Gamma_r, \Gamma_o$);
18:        if (flag = 1), return $(1, q_{e1})$;
19:      end if
20:    end for
21:  end for
22: end if-else
23: return $(0, q_e)$;
```

Lemma 2: Algorithm II is of polynomial complexity.

Proof: The complexities of the for-loops in Lines 2–4, Lines 5–7, Lines 8–10, and Lines 11–22 are $O(s_1 \times m)$, $O(s_1)$, $O(s_1 \times m)$, and $O(s_1 \times m)$, respectively, where $s_1 = \Sigma_{i\in Z_n}|J_i|$. Thus the complexity of Algorithm II is $O(s_1 \times m)$. ■

Theorem 1: Algorithm III is of polynomial complexity.

Proof: The for-loop in Lines 6–21 can be iterated at most $s_1 = \Sigma_{i\in Z_n}|J_i|$ times. The complexity of an iteration is dominated by the for-loop in Lines 13–20, which can be iterated at most $s_3 = \max\{|J_i|: i \in Z_n\}$ times. The complexity of an iteration in such a loop is O(Algorithm I) + O(Algorithm II) = $O((s_1 + s_2) \times s_1 \times m)$. Thus the complexity of Algorithm III is $O(s_1 \times s_3 \times (s_1 + s_2) \times s_1 \times m) = O(s_1^2 \times (s_1 + s_2) \times s_3 \times m)$. ■

Theorem 2: $Q_R(q_0, \text{MBA}_\text{Y}) \subseteq Q_R(q_0, \text{MBA}_\text{L})$.

Proof: Given $q \in Q_R(q_0, \text{MBA}_\text{Y})$, let $(\Lambda_a, \Lambda_r, \Lambda_o)$ = Algorithm II(q). Since $q \in Q_R(q_0, \text{MBA}_\text{Y})$, $\text{MBA}_1(q, \Lambda_a, \Lambda_r, \Lambda_o) = (1, q_1)$, where $q_1 \in Q$ is the output state. Since $\text{MBA}_1(q, \Lambda_a, \Lambda_r, \Lambda_o) = (1, q_1)$, $q \in Q_R(q_0, \text{MBA}_\text{L})$. ■

Remark 1: By Theorem 2, all markings reachable under MBA_Y from q_0 are reachable under MBA_L from q_0.

C. Failure Dependent Subnet and Its PN-based DAP

MBA_L can ensure that all jobs in $\mathbb{S}$ can be moved out of $\mathbb{S}$ or into the first-met FD resources in their remaining routes. However, it cannot ensure that those jobs remaining in the system after such movements can be moved out of $\mathbb{S}$. To see this, consider a state of the system in Fig. 1, i.e., $q = 1 + 1 + x_{31} + x_{41}$. Obviously, $\text{MBA}_\text{L}(q) = (1, q_e)$, where $q_e = 1 + 1 + x_{32} + x_{41}$. At q_e, a job is in r_5 and waiting for being produced by r_6, while a job is in r_6 and waiting for being produced by r_5. Thus, a deadlock occurs at q_e. To avoid such deadlocks, a PN-based DAP is introduced.

Given a state $q \in Q_R(q_0, \text{MBA}_\text{L})$, after executing MBA_L on it, all jobs remaining in the system are in the buffers of FD resources. To ensure the resulting state is deadlock-free, we only need to ensure the safety of a subsystem that consists of FD resources. Thus, we reduce the PN model of an AMS below.

1) Failure Dependent Subnet (FDS)

Let $\Omega_F = \{p_{ij}: \rho(p_{ij}) \in R_F\}$ be the set of operation places where jobs occupy FD resources. Given $p_{ij} \in \Omega_F$, let $\Omega(p_{ij}) = \{p_{ij}\} \cup \{p_{ic}: c < j \text{ and } \forall d \in [c, j), \rho(p_{jd}) \notin R_F\}$. Let $P_f = \cup_{p_{ij} \in \Omega_F}\Omega(p_{ij})$ and $\Omega(r) = \cup_{p_{ij} \in H(r)}\Omega(p_{ij})$ for $r \in R_F$.

Definition 5: The FDS of $N = (P \cup P_R, T, F)$ is a PN $N_f = (P_f \cup R_F, T_f, F_f)$, which is obtained by the following three steps:

1) Delete all resource places except for those in R_F and those arcs connected to them from N.

2) Delete all operation places except for those in P_f, all arcs connected to these places, and all isolated transitions.

3) For each $p_{ij} \in \Omega_F$ with $|\Omega(p_{ij})| > 1$, delete (r, t_{ij}), where $t_{ij} = {}^\bullet p_{ij}$ and $r = \rho(p_{ij})$. Let p_{ik} be the operation place in $\Omega(p_{ij})$ with the smallest index and $t_{ik} = {}^\bullet p_{ik}$. If $(t_{ik}, r) \notin F$, then add (r, t_{ik}); otherwise, delete (t_{ik}, r).

Theorem 3: $\forall J_{kt} \in \mathbb{J}$, if $\exists r \in \{R^U \cap \vartheta_{kt}\}$, then $\forall r_1 \in \{R^U \backslash \{r\}\}$, $\vartheta_{kt} \cap R_F(r_1) = \emptyset$.

Proof: Suppose that $\exists r_1 \in \{R^U \backslash \{r\}\} \ni \vartheta_{kt} \cap R_F(r_1) \neq \emptyset$ and $\rho(J_{k(t+c)}) \in R_F(r_1)$, where $c \in Z^+$. Since $\rho(J_{k(t+c)}) \in R_F(r_1)$, $\exists d \in Z \ni \rho(J_{k(t+c+d)}) = r_1$. Since $\exists r \in \{R^U \cap \vartheta_{kt}\}$, $\exists e \in Z^+ \ni \rho(J_{k(t+e)}) = r \neq r_1$. It is a contradiction to the hypothesis that a job requires no more than one unreliable resource. ■

Remark 2: By Theorem 3, if a job requires no more than one unreliable resource, then all sets of FD resources on unreliable resources are nonintersecting.

$\forall r_i \in R^U$, let $N_{fi} = (P_{fi} \cup R_F(r_i), T_{fi}, F_{fi})$ be the subnet corresponding to r_i, where $P_{fi} = \{\Omega(p_{st}): \rho(p_{st}) \in R_F(r_i)\}$, $T_{fi} = \{{}^\bullet p_{st}, p_{st}{}^\bullet: p_{st} \in P_{fi}\}$, $F_{fi} = \{(p, t), (t, p) \in F_f: p \in P_{fi} \cup R_F(r_i), t \in T_{fi}\}$.

Corollary 1: $\forall r_i, r_j \in R^U$, N_{fi} is independent of N_{fj}.

Proof: By Theorem 3, $R_F(r_i) \cap R_F(r_j) = \emptyset$. By the definition of $\Omega(p)$ for $p \in \Omega_F$, $\forall p_1, p_2 \in \Omega_F$, $\Omega(p_1) \cap \Omega(p_2) = \emptyset$. Thus, $P_{fi} \cap P_{fj} = \emptyset$, $T_{fi} \cap T_{fj} = \emptyset$, and $F_{fi} \cap F_{fj} = \emptyset$. ■

Remark 3: By Corollary 1, N_f is the combination of all N_{fi}, where $r_i \in R^U$.

Example 3: Consider the system in Example 1, $\Omega_F = \{p_{24}, p_{32}\text{-}p_{34}, p_{41}, p_{42}, p_{44}\}$, $\Omega(p_{24}) = \{p_{21}\text{–}p_{24}\}$, $\Omega(p_{32}) = \{p_{31}, p_{32}\}$, $\Omega(p_{33}) = \{p_{33}\}$, $\Omega(p_{34}) = \{p_{34}\}$, $\Omega(p_{41}) = \{p_{41}\}$, $\Omega(p_{42}) = \{p_{42}\}$, and $\Omega(p_{44}) = \{p_{43}, p_{44}\}$. To derive its FDS, we first delete resource places r_1, r_2, and r_3, and arcs related with them from Fig. 1. Then, we delete operation places p_{11}, p_{12}, and p_{13}, arcs related with them, and isolated transitions t_{11}, t_{12}, t_{13}, and t_{14}. Finally, delete arcs (r_4, t_{24}), (r_5, t_{32}), and (r_7, t_{44}), and add arcs (r_4, t_{21}), (r_5, t_{31}), and (r_7, t_{43}). The FDS of the system is exhibited in Fig. 2, which is the composition of N_{f4} and N_{f7}.

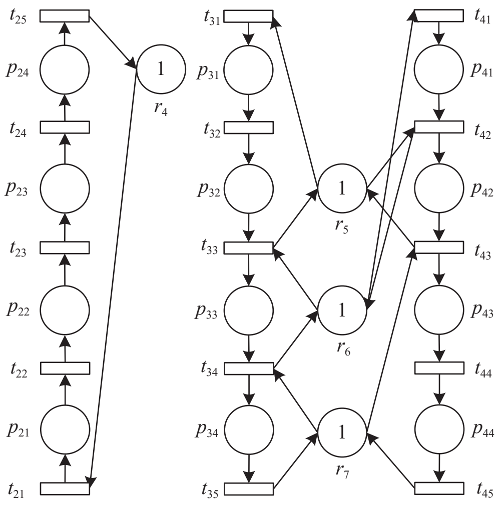


Fig. 2. FDS of the AMS in Example 1.

Given $M \in \mathbb{R}(N, M_0)$, let $\gamma(M) = M_1$ denote the marking of N_f corresponded to M, which is defined below. $\forall p \in P_f$, $M_1(p) = M(p)$, and $\forall r \in R_F$, $M_1(r) = C(r) - M(\Omega(r))$. Specially, let $M_{f0} = \gamma(M_0)$ denote the initial marking of N_f.

2) PN-based DAP for FDS

Let $p_{ij} \in \Omega_F$ be an operation place in N_f with $|\Omega(p_{ij})| > 1$. In N_f, all places in $\Omega(p_{ij})$ use $\rho(p_{ij})$. By treating them as one operation place, FDS belongs to the class of S^3PR [3]. Thus, the DAP in [16] can be adopted to ensure FDS deadlock-free.

Xing *et al.* [16] characterize deadlocks in S^3PR as saturated maximal prefect resource-transition circuits (MPRT-circuits). If a resource r belongs to two MPRT-circuits that do not include each other and $C(r) = 1$, then r is called as a center resource. For S^3PR without center resources, Xing *et al.* [16] develop a one-step-look-ahead deadlock search (DS) algorithm, which is proved to be optimal and $O(\text{DS}) = O(s_4)$, where $s_4 = \Sigma_{r \in R_F}C(r)$. For S^3PR with ξ-resources, Xing *et al.* [16] reduce them to obtain a reduced net without center resources. Then, by applying DS to the reduced net and using the controlled reduced net as a supervisor for the original net, a suboptimal polynomial controller for the original net is obtained. For simplicity, let DS be the DAP for N_f with or without center resources.

Remark 4: Since N_f is the composition of all N_{fi}, where $r_i \in R^U$, DS can be treated as the composition of all DS_i, where DS_i is a DAP for N_{fi}.

Given $q \in Q$, q is admitted by DS if $\gamma(\chi(q)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, where $\mathbb{R}(N_f, M_{f0}, \text{DS})$ is the set of all markings of N_f reachable from M_{f0} under DS.

Given a state q that is admitted by DS, let $\pi \in \xi(q)$ be an event enabled at q, and $q_1 = \delta(q, \pi)$. Now, we have the following three lemmas.

Lemma 3: If $\pi \in \Sigma_u$, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Proof: By the definition of χ, we know that $\chi(q) = \chi(q_1)$ if $\pi \in \Sigma_{u1}$ or Σ_{u2}. Since $\gamma(\chi(q)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. ■

Remark 5: By Lemma 3, the occurrences of uncontrollable events are not prohibited by DS.

Lemma 4: If $\pi = \alpha_{ij}$ is a controllable event, and the transition corresponding to α_{ij}, i.e., $t_{ij} \notin T_f$, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Proof: Since $t_{ij} \notin T_f$, the occurring of α_{ij} does not change $M(p)$ $\forall p \in P_f$, where $M = \gamma(\chi(q))$. By the definition of γ, we know that $\gamma(\chi(q)) = \gamma(\chi(q_1))$. Since $\gamma(\chi(q)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. ■

Lemma 5: If $\pi = \alpha_{ij}$ is a controllable event, the transition corresponding to α_{ij}, i.e., $t_{ij} \in T_f$, and ${}^{(r)}t_{ij} = \emptyset$ in N_f, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Proof: Since $t_{ij} \in T_f$, and ${}^{(r)}t_{ij} = \emptyset$, the occurring of α_{ij} which is corresponded to the firing of t_{ij} does not increase $M(r)$, $\forall r \in R_F$, where $M = \gamma(\chi(q))$. Thus, $\forall r \in R_F$, $M(r) \geq M_1(r)$, where $M_1 = \gamma(\chi(q_1))$. Since $M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, $M_1 \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. ■

Let Ξ_1 and Ξ_2 denote the set of all controllable events described in Lemmas 4 and 5, respectively, and $\Xi_3 = \Sigma_c \backslash \{\Xi_1 \cup \Xi_2\}$.

Example 4: Consider the system in Example 1, $T = \{t_{11}–t_{14}, t_{21}–t_{25}, t_{31}–t_{35}, t_{41}–t_{45}\}$, $T_f = \{t_{21}–t_{25}, t_{31}–t_{35}, t_{41}–t_{45}\}$. Since $T \backslash T_f = \{t_{11}–t_{14}\}$, $\Xi_1 = \{\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}\}$. Since ${}^{(r)}t_{22} = {}^{(r)}t_{23} = {}^{(r)}t_{24} = {}^{(r)}t_{32} = {}^{(r)}t_{44} = \emptyset$, $\Xi_2 = \{\alpha_{22}–\alpha_{25}, \alpha_{32}, \alpha_{35}, \alpha_{44}, \alpha_{45}\}$. $\Xi_3 = \{\alpha_{11}–\alpha_{14}, \alpha_{21}–\alpha_{25}, \alpha_{31}–\alpha_{35}, \alpha_{41}–\alpha_{45}\} \backslash \{\Xi_1 \cup \Xi_2\} = \{\alpha_{21}, \alpha_{31}, \alpha_{33}, \alpha_{34}, \alpha_{41}–\alpha_{43}\}$.

D. Proposed Robust DAP

Our policy for $\mathbb{S} = \{R, J, Q, \Sigma, \xi, \delta\}$ is a function Δ_{LLXW}: $Q \times \Sigma \rightarrow \{0, 1\}$ defined below. Let $q \in Q_R(q_0, \Delta_{\text{LLXW}})$ be a state and $\pi \in \xi(q)$ be an enabled event to be judged, and $q_1 = \delta(q, \pi)$. If $\pi \in \Sigma_u$, then $\Delta_{\text{LLXW}}(q, \pi) = 1$. Else if $\pi \in \{\Xi_1 \cup \Xi_2\}$ and $\text{MBA}_{\text{L}}(q_1) = (1, q_e)$, then $\Delta_{\text{LLXW}}(q, \pi) = 1$. Else, let $\text{MBA}_{\text{L}}(q) = (1, q_e)$, $M = \gamma(\chi(q_e))$, and t be the transition corresponded to π. $\Delta_{\text{LLXW}}(q, \pi) = 1$ if $\text{DS}(M, t)$ = permitted and $\text{MBA}_{\text{L}}(q_1) = (1, q_{e1})$.

Next, we prove that Δ_{LLXW} satisfies the three properties in Definition 2.

Lemma 6: Given $q \in Q_R(q_0, \Delta_{\text{LLXW}})$, let $\text{MBA}_{\text{L}}(q) = (1, q_e)$ and σ be a sequence of events such that $q = \delta(q_0, \sigma)$, then $M = \gamma(\chi(q_e)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Proof: We prove $M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$ by mathematical induction over $|\sigma|$ as below.

If $|\sigma| = 0$, then $q = \delta(q_0, \sigma) = q_0$. Thus, $\text{MBA}_{\text{L}}(q) = \text{MBA}_{\text{L}}(q_0) = (1, q_0)$, and $M = \gamma(\chi(q_0)) = M_{f0} \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Suppose that $|\sigma| = n > 0$, $M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. Let $\pi \in \xi(q)$ be an event enabled at q, and $q_1 = \delta(q_0, \sigma\pi)$. Suppose that $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$. Let $\text{MBA}_{\text{L}}(q_1) = (1, q_{e1})$ and $M_1 = \gamma(\chi(q_{e1}))$. Now, we prove $M_1 \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Case 1: $\pi = \beta_{st} \in \Sigma_{u1}$. If $J_{st} \in \mathbb{J}^1(q) \cup \mathbb{J}^3(q) \cup \mathbb{J}^4(q)$, by MBA_{L}, $q_e = q_{e1}$, thus $M_1 = M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. If $J_{st} \in \mathbb{J}^2(q)$, the job corresponding to β_{st} will be moved out of $\mathbb{S}$ by MBA_{L} at q_1. Since such a job does not complete its service on $\rho(J_{st})$, it will not be moved by MBA_{L} at q. Thus, all elements in q_{e1} are equal to those in q_e except for $q_{e1}(y_{st}) = q_e(y_{st}) - 1$. Thus, $M_1(p) = M(p)$, $\forall p \in \{P_f \backslash \{p_{st}\}\}$, and $M_1(p_{st}) = M(p_{st}) - 1$, where p_{st} is the operation place corresponding to J_{st}. Since $M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, $M_1 \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Case 2: $\pi \in \Sigma_{u2}$. By MBA_{L}, $q_e = q_{e1}$, thus $M_1 = M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Case 3: $\pi = \alpha_{ik} \in \Xi_1$. Since $\alpha_{ik} \in \Xi_1$, $J_{i(k-1)} \in \mathbb{J}^1(q)$. If $k \leq |J_i|$, $J_{ik} \in \mathbb{J}^1(q_1)$ and $\{\mathbb{J}^1(q) \cup \{J_{ik}\}\} = \{\mathbb{J}^1(q_1) \cup \{J_{i(k-1)}\}\}$, $\mathbb{J}^2(q) = \mathbb{J}^2(q_1)$, and $\mathbb{J}^3(q) = \mathbb{J}^3(q_1)$. Otherwise, $\mathbb{J}^1(q) = \{\mathbb{J}^1(q_1) \cup \{J_{i(k-1)}\}\}$, $\mathbb{J}^2(q) = \mathbb{J}^2(q_1)$, and $\mathbb{J}^3(q) = \mathbb{J}^3(q_1)$. Since the jobs at stage $J_{i(k-1)}$ or J_{ik} will be moved out of $\mathbb{S}$ by MBA_{L}, $q_e = q_{e1}$ and $M_1 = M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Case 4: $\pi = \alpha_{ik} \in \Xi_2$. Since $\alpha_{ik} \in \Xi_2$, $J_{i(k-1)} \in \{\mathbb{J}^2(q) \cup \mathbb{J}^3(q)\}$. If $J_{i(k-1)} \in \mathbb{J}^2(q)$ and $k \leq |J_i|$, $J_{ik} \in \mathbb{J}^1(q_1)$ and $\mathbb{J}^1(q_1) = \mathbb{J}^1(q) \cup \{J_{ik}\}$, $\mathbb{J}^2(q_1) \cup \{J_{i(k-1)}\} = \mathbb{J}^2(q)$, and $\mathbb{J}^3(q_1) = \mathbb{J}^3(q)$. If $J_{i(k-1)} \in \mathbb{J}^2(q)$ and $k > |J_i|$, $\mathbb{J}^1(q) = \mathbb{J}^1(q_1)$, $\mathbb{J}^2(q) = \mathbb{J}^2(q_1) \cup \{J_{i(k-1)}\}$, and $\mathbb{J}^3(q) = \mathbb{J}^3(q_1)$. Since the job at stage $J_{i(k-1)}$ or J_{ik} will be moved out of $\mathbb{S}$ by MBA_{L}, $q_e = q_{e1}$, and $M_1 = M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

If $J_{i(k-1)} \in \mathbb{J}^3(q)$, $J_{ik} \in \mathbb{J}^3(q_1)$ and $\mathbb{J}^1(q_1) = \mathbb{J}^1(q)$, $\mathbb{J}^2(q_1) = \mathbb{J}^2(q)$, and $\mathbb{J}^3(q_1) \cup \{J_{i(k-1)}\} = \mathbb{J}^3(q) \cup \{J_{ik}\}$. Since $\alpha_{ik} \in \Xi_2$, the first-met FD resource in $\vartheta_{i(k-1)}$ is the same as that in ϑ_{ik}. Since the job at stage $J_{i(k-1)}$ or J_{ik} will be moved into the same FD resource, $q_e = q_{e1}$, and $M_1 = M \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Case 5: $\pi = \alpha_{ik} \in \Xi_3$. Let $t_{jk} \in T_f$ be the transition corresponding to α_{ik}. Since $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$, $\text{DS}(M, t_{jk})$ = permitted. Thus, $M[t_{jk}> = M_2 \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. Since $\alpha_{ik} \in \Xi_3$, the occurrence of α_{ik} stands for moving a raw job that requires FD resources into the system or moving a job out of an FD resource. MBA_{L} will move such a job into the first-met FD resource in ϑ_{ik}. Let $\sigma_1 = \alpha_{i(k+1)}\alpha_{i(k+2)}\ldots\alpha_{i(k+c)}$ be the sequence of controllable events corresponding to such a movement and $T_1 = t_{i(k+1)}t_{i(k+2)}\ldots t_{i(k+c)}$ be the sequence of transitions corresponding to σ_1. Then, $q_{e1} = \delta(q_e, \alpha_{ik}\sigma_1)$ and $M_2[T_1> M_1 = \gamma(\chi(q_{e1}))$. By Lemma 5, the firings of transitions in T_1 from M_2 are permitted by DS, $M_1 \in \mathbb{R}(N_f, M_{f0}, \text{DS})$.

Lemma 7: $\forall q \in Q_R(q_0, \Delta_{\text{LLXW}})$, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q, \sigma) = q_1 \ni \mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$ and $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. ■

Proof: Recall that $q \in Q_R(q_0, \text{MBA}_{\text{L}})$ only if 1) all jobs at stage $J_{ij} \in \mathbb{J}^1(q)$ can be moved out of $\mathbb{S}$; 2) all jobs at stage $J_{ij} \in \mathbb{J}^2(q)$ that have finished their current operations on $\rho(J_{ij})$ can be moved out of $\mathbb{S}$; and 3) all jobs at stage $J_{ij} \in \mathbb{J}^3(q)$ can be moved into the first-met FD resource in ϑ_{ij}. If $J_{ij} \in \mathbb{J}^1(q)$, every event π corresponding to the movement in 1) is in $\{\Sigma_{u1} \cup \Xi_1\}$. If $J_{ij} \in \mathbb{J}^2(q)$, every event π corresponding to the movement in 2) is in $\{\Sigma_{u1} \cup \Xi_1 \cup \Xi_2\}$. If $J_{ij} \in \mathbb{J}^3(q)$, every event π corresponding to the movement in 3) is in $\{\Sigma_{u1} \cup \Xi_2\}$. Since all these events do not contain the completion of service events on unreliable resources, they are executable no matter whether unreliable resources are operational or failed. By Lemmas 3–5, all these events are permitted by DS. Thus, $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$. Since $\mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$, $\text{MBA}_\text{L}(q_1) = (1, q_1)$. By Lemma 6, $\gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. ■

Lemma 8: $\forall q \in Q \ni \mathbb{J}^1(q) = \mathbb{J}^2(q) = \mathbb{J}^3(q) = \emptyset$ and $M = \gamma(\chi(q)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, if $\exists r_i \in R^U \ni sv_i = 1$ and $M(P_{fi}) > 0$, then $\exists t_{jk} \in T_{fi} \ni \text{DS}(M, t_{jk})$ = permitted and $q_1 = \delta(q, \alpha_{jk}) \in Q_R(q_0, \text{MBA}_\text{L})$.

Proof: By Remark 4, DS is the composition of all DS_i, $\forall r_i \in R^U$. Since DS_i is a DAP for N_{fi} and $M(P_{fi}) > 0$, at least one transition $t_{jk} \in T_{fi}$ is enabled at M.

Since $\mathbb{J}^1(q) = \mathbb{J}^2(q) = \mathbb{J}^3(q) = \emptyset$, all jobs are in FD resources and all $R\backslash R_F$ are idle at q. Now, we consider the following four cases.

Case 1: $k \le |J_j|$, $\rho(J_{jk}) \in R\backslash R_F$, and $t_{j(k+1)} \in T_{fi}$. Since all $R\backslash R_F$ are idle at q, $\alpha_{jk} \in \xi(q)$. Since $t_{j(k+1)} \in T_{fi}$, the job moved by event α_{jk} will be further moved to an FD resource by MBA_L. All resources required by such a movement are in $R\backslash R_F$. Since all required resources are idle, $q_1 = \delta(q, \alpha_{jk}) \in Q_R(q_0, \text{MBA}_\text{L})$.

Case 2: $k \le |J_j|$, $\rho(J_{jk}) \in R\backslash R_F$, and $t_{j(k+1)} \notin T_{fi}$. Since all $R\backslash R_F$ are idle at q, $\alpha_{jk} \in \xi(q)$. Since $t_{j(k+1)} \notin T_{fi}$, the job moved by event α_{jk} will be moved out of $\mathbb{S}$ by MBA_L. All resources required by such a movement are in $R\backslash R_F$. Since all required resources are idle, $q_1 = \delta(q, \alpha_{jk}) \in Q_R(q_0, \text{MBA}_\text{L})$.

Case 3: $k \le |J_j|$ and $\rho(J_{jk}) \in R_F$. Let $\rho(J_{jk}) = {}^{(r)}t_{jk}$. Since $\text{DS}(M, t_{jk})$ = permitted, $\rho(J_{jk})$ has at least one idle buffer, and hence $\alpha_{jk} \in \xi(q)$. Since $\rho(J_{jk}) \in R_F$, $\mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$, thus $q_1 \in Q_R(q_0, \text{MBA}_\text{L})$.

Case 4: $k = |J_j| + 1$. Since $\text{DS}(M, t_{jk})$ = permitted, there are some finished parts at $J_{j(k-1)}$, and hence $\alpha_{jk} \in \xi(q)$. Since $k = |J_j| + 1$, $\mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$, thus $q_1 \in Q_R(q_0, \text{MBA}_\text{L})$. ■

Remark 6: By Lemma 7, Δ_{LLXW} guarantees that all jobs can be moved either out of the system or to the first-met FD resources in their remaining routes. By Lemma 8, if those jobs in FD resources do not require failed resources, they can be moved out of the system under Δ_{LLXW}.

Theorem 4: $\forall q \in Q_R(q_0, \Delta_{\text{LLXW}})$ with a set of failed resources, i.e., $\mathcal{R}(q)$, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q, \sigma) = q_1 \ni \forall J_{ik} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q)\}\}$, $q_1(x_{ik}) + q_1(y_{ik}) = 0$.

Proof: By Lemma 7, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q, \sigma) = q_1 \ni \mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$ and $M = \gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$. Thus, we only need to prove that all jobs at $J_{ik} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q)\}\}$ can be moved out of $\mathbb{S}$. Since $\mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1) = \emptyset$, and $M = \gamma(\chi(q_1)) \in \mathbb{R}(N_f, M_{f0}, \text{DS})$, by Lemma 8, if $\exists r_u \in \{R^U\backslash\mathcal{R}(q)\}$ and $M(P_{fu}) > 0$, then $\exists t_{ik} \in T_{fu} \ni \text{DS}(M, t_{ik})$ = permitted and $q_1 = \delta(q, \alpha_{ik}) \in Q_R(q_0, \text{MBA}_\text{L})$. Since $\mathbb{J}^1(q_1) = \mathbb{J}^2(q_1) = \mathbb{J}^3(q_1)$, $\text{MBA}_\text{L}(q_1) = (1, q_1)$. Since $\text{DS}(M, t_{ik})$ = permitted and $q_1 = \delta(q, \alpha_{ik}) \in Q_R(q_0, \text{MBA}_\text{L})$, $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$. Thus, the aforementioned process can be repeated until all jobs at stage $J_{ik} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q)\}\}$ have been moved out of $\mathbb{S}$. ■

Remark 7: By Theorem 4, Δ_{LLXW} guarantees that all jobs not requiring failed resource can be completed.

Corollary 2: $\forall q \in Q_R(q_0, \Delta_{\text{LLXW}})$, all jobs not requiring resources in $\mathcal{R}(q)$ can be processed continuously if no additional failures or repairs occur.

Proof: By Theorem 4, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q, \sigma) = q_1 \ni \forall J_{ik} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q)\}\}$, $q_1(x_{ik}) + q_1(y_{ik}) = 0$. At q_1, all resources except for those in $\{R_F(r): r \in \mathcal{R}(q)\}$ are idle. Thus, moving a job that does not require those resources in $\{R_F(r): r \in \mathcal{R}(q)\}$ is permitted by MBA_L and DS. Let $q_2 \in Q_R(q_0, \Delta_{\text{LLXW}})$ be the state obtained after executing such a movement from q_1. By Theorem 4, such a job can be completed and moved out of $\mathbb{S}$. The aforementioned process can be repeated if no additional failures or repairs occur. ■

Theorem 5: $\forall q \in Q_R(q_0, \Delta_{\text{LLXW}})$, if $\kappa_i \in \xi(q)$, $\delta(q, \kappa_i) = q_1$ is an FI state.

Proof: Since $\kappa_i \in \Sigma_u$, $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$. By Theorem 4, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q_1, \sigma) = q_2 \ni \forall J_{jk} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q_1)\}\}$, $q_2(x_{jk}) + q_2(y_{jk}) = 0$. Thus, q_1 is an FI state. ■

Theorem 6: $\forall q \in Q_R(q_0, \Delta_{\text{LLXW}})$, if $\eta_i \in \xi(q)$, $\delta(q, \eta_i) = q_1$ is an FI state.

Proof: Since $\eta_i \in \Sigma_u$, $q_1 \in Q_R(q_0, \Delta_{\text{LLXW}})$. By Theorem 4, $\exists \sigma \in \{\Sigma_c \cup \Sigma_{u1}\}^* \ni \delta(q_1, \sigma) = q_2 \ni \forall J_{jk} \notin \{J_{st} \in \mathbb{J}: \rho(J_{st}) \in \{R_F(r): r \in \mathcal{R}(q_1)\}\}$, $q_2(x_{jk}) + q_2(y_{jk}) = 0$. Thus, q_1 is an FI state. ■

Theorem 7: Δ_{LLXW} is a robust DAP.

Proof: Corollary 2 and Theorems 5 and 6 prove that Δ_{LLXW} has properties 1–3 in Definition 2, respectively. ■

Remark 8: In the aforementioned process of proof, DS is merely treated as a DAP. It means that if there is another DAP that is better than DS, then it can be integrated with MBA_L to make a new robust DAP for the studied systems.

IV. Comparison Studies

In this section, we compare Δ_{LLXW} with those controllers for AMSs with $|R^U| = 1$ and those controllers for AMSs with $|R^U| > 1$, respectively. To be fair, $|Q_R(q_0, \Delta_{\text{LLXW}})|$ are computed under the assumption that unreliable resources can only fail when they are working.

Consider an AMS with $R^R = \{r_1, r_3, r_4, r_5, r_6\}$, $R^U = \{r_2\}$, $C(r_6) = 2$, $C(r_1) = C(r_2) = C(r_3) = C(r_4) = C(r_5) = 1$, and $J = \{J_1, J_2, J_3, J_4\}$. $|J_1| = |J_2| = |J_3| = 3$, $|J_4| = 4$, $\rho(J_{22}) = \rho(J_{42}) = r_1$, $\rho(J_{23}) = \rho(J_{44}) = r_2$, $\rho(J_{13}) = \rho(J_{31}) = r_3$, $\rho(J_{21}) = \rho(J_{43}) = r_4$, $\rho(J_{11}) = \rho(J_{33}) = \rho(J_{41}) = r_5$, and $\rho(J_{12}) = \rho(J_{32}) = r_6$. Seven controllers proposed in [23] (Δ_{L1}), [24] (Δ_{L2}), [28] (Δ_{L3}), [29] (Δ_{L4}), [31] (Δ_{W1}), [32] (Δ_{W2}), and [36] (Δ_{Y1}) can be used for it. In [28], we have proved that Δ_{L3} is more permissive than Δ_{L2} and Δ_{Y1}. Besides, in [29], we have proved that Δ_{L4} admits more states than Δ_{L1} does. Therefore, we only need to compare Δ_{LLXW} with Δ_{L3}, Δ_{L4}, Δ_{W1}, and Δ_{W2}. $|Q_R(q_0, \Delta_{\text{LLXW}})|$ = 32 039, $|Q_R(q_0, \Delta_{\text{L3}})|$ = 29 659, $|Q_R(q_0, \Delta_{\text{L4}})|$ = 19 075, $|Q_R(q_0, \Delta_{\text{W1}})|$ = 835, and $|Q_R(q_0, \Delta_{\text{W2}})|$ = 32 039. Obviously, Δ_{LLXW} admits many more states than Δ_{L3}, Δ_{L4}, and Δ_{W1} do. Although Δ_{LLXW} and Δ_{W2} admit the same number of states, Δ_{LLXW} can handle AMS with $|R^U| > 1$.

The illustrative AMS with $|R^U| > 1$ is shown in Example 1. Five controllers proposed in [18] (Δ_{C1}), [20] (Δ_{C2}), [31] (Δ_{W1}), [34] (Δ_{Y2}), and [35] (Δ_{Y3}) can be used for it. Yue *et al.* [34] have proved that Δ_{Y2} admits more states than Δ_{C1} does.

Therefore, we only compare $\Delta_{\rm LLXW}$ with $\Delta_{\rm C2}$, $\Delta_{\rm W1}$, $\Delta_{\rm Y2}$, and $\Delta_{\rm Y3}$. $|Q_R(q_0, \Delta_{\rm LLXW})| = 73\,572$, $|Q_R(q_0, \Delta_{\rm C2})| = 22\,662$, $|Q_R(q_0, \Delta_{\rm W1})| = 3085$, $|Q_R(q_0, \Delta_{\rm Y2})| = 35\,628$, $|Q_R(q_0, \Delta_{\rm Y3})| = 5106$. Obviously, $\Delta_{\rm LLXW}$ admits many more states than $\Delta_{\rm C2}$, $\Delta_{\rm W1}$, $\Delta_{\rm Y2}$, and $\Delta_{\rm Y3}$ do. Moreover, $\Delta_{\rm LLXW}$ can ensure the robustness of the AMSs where unreliable resources can fail when they are idle.

There are three main reasons why $\Delta_{\rm LLXW}$ admits more states than $\Delta_{\rm L3}$, $\Delta_{\rm L4}$, $\Delta_{\rm W1}$, $\Delta_{\rm C2}$, $\Delta_{\rm Y2}$ and $\Delta_{\rm Y3}$. At first, DS is optimal for N_f without ξ-resources, while the neighborhood constraints in $\Delta_{\rm W1}$, $\Delta_{\rm W2}$, $\Delta_{\rm C2}$, $\Delta_{\rm Y2}$ and $\Delta_{\rm Y3}$ are suboptimal. Secondly, $\text{MBA}_{\rm L}$ admits more states than the $\text{MBA}_{\rm Y}$ in $\Delta_{\rm L4}$ and $\Delta_{\rm Y2}$. Thirdly, $\text{MBA}_{\rm L}$ tries to move every job that has finished its operation on an unreliable resource out of $\mathbb{S}$ to release the occupied buffer, while these jobs are not moved by $\Delta_{\rm L3}$, $\Delta_{\rm W1}$, $\Delta_{\rm W2}$, $\Delta_{\rm C2}$ and $\Delta_{\rm Y3}$. Since these policies are so different, comparing them theoretically is still an open work. Besides, $\Delta_{\rm Y3}$ is also applicable for AMSs where a job can require multiple unreliable resources. Extending $\Delta_{\rm LLXW}$ for this kind of systems is in our research agenda.

V. Conclusion

This paper proposes a polynomial robust DAP for AMSs with unreliable resources, which consists of $\text{MBA}_{\rm L}$ and a DAP for the FDS of the system. $\text{MBA}_{\rm L}$ can ensure that the jobs not requiring unreliable resources can be moved out of the system and those requiring unreliable resources can be moved into the first-met FD resources in their remaining routes. Also, it can guarantee that all jobs that have finished their operations on unreliable resources can be moved out of the system. The DAP can guarantee that all jobs that are in the system after executing $\text{MBA}_{\rm L}$ and do not require failed resources can be moved out of the system. $\text{MBA}_{\rm L}$ together with the DAP is proved to be robust to the failures of unreliable resources. Comparison results with existing policies show that our policy is more permissive than existing ones. Moreover, it can handle the AMS where unreliable resources may fail when they are idle or working. Proposing an optimal robust DAP for the studied AMS is hard but significant, which needs to be studied further. Another one is to propose a robust DAP for those AMSs with multiply unreliable resource requisitions.

References

[1] Y. F. Chen and Z. W. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, no. 5, pp. 1028–1034, Mar. 2011.

[2] Y. F. Chen, Z. W. Li, and M. C. Zhou, "Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern., A, Syst., Humans*, vol. 42, no. 42, pp. 615–629, May 2012.

[3] 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.

[4] M. P. Fanti and M. C. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern., A, Syst., Humans*, vol. 34, no. 1, pp. 5–22, Jan. 2004.

[5] H. S. Hu, M. C. Zhou, Z. W. Li, and Y. Tang, "Deadlock-free control of ams with flexible routes and assembly operations using Petri nets," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 109–121, Feb. 2013.

[6] H. S. Hu and M. C. Zhou, "A Petri net-based discrete event control of automated manufacturing systems with assembly operations," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 2, pp. 513–524, Mar. 2015.

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

[8] H. X. Liu, K. Y. Xing, M. C. Zhou, L. B. Han, and F. Wang, "Transition cover-based design of Petri net controllers for automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 2, pp. 196–208, Feb. 2014.

[9] H. X. Liu, W. Wu, H. Su, and Z. Zhang, "Design of optimal Petri net controllers for a class of flexible manufacturing systems with key resources," *Inform. Sciences*, vol. 363, pp. 221–234, Oct. 2016.

[10] G. Y. Liu and K. Barkaoui, "Necessary and sufficient liveness condition of GS3PR Petri nets," *Int. J. Syst. Science*, vol. 46, no. 7, pp. 1147–1160, May 2015.

[11] G. Y. Liu and D. Y. Chao, "Further reduction of minimal first-met bad markings for the computationally efficient synthesis of a maximally permissive controller," *Int. J. Control*, vol. 88, no. 8, pp. 1–6, Aug. 2015.

[12] J. C. Luo, Z. Q. Liu, and M. C. Zhou, "A Petri net-based deadlock avoidance policy for flexible manufacturing systems with assembly operations and multiple resource acquisition," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3379–3387, Jun. 2019.

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

[14] N. Q. Wu, M. C. Zhou, and G. Hu, "Petri net modeling and one-step look-ahead maximally permissive deadlock control of automated manufacturing systems," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 1, pp. 1–10, Jan. 2013.

[15] N. Q. Wu, M. C. Zhou, and Z. W. 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.

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

[17] K. Y. Xing, F. Wang, M. C. Zhou, H. Lei, and J. C. Luo, "Deadlock characterization and control of flexible assembly systems with Petri nets," *Automatica*, vol. 87, pp. 358–364, Jan. 2018.

[18] S. F. Chew and M. A. Lawley, "Robust supervisory control for production systems with multiple resource failures," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, pp. 309–323, Jul. 2006.

[19] 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.

[20] S. F. Chew, S. Y. Wang, and M. A. Lawley, "Resource failure and blockage control for production systems," *Int. J. Comput. Integ. M.*, vol. 24, no. 3, pp. 229–241, 2011.

[21] F. Hsieh, "Fault-tolerant deadlock avoidance algorithm for assembly processes," *IEEE Trans. Syst., Man, Cybern., A, Syst., Humans*, vol. 34, no. 1, pp. 65–79, Jan. 2004.

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

[23] M. A. Lawley, "Control of deadlock and blocking for production systems with unreliable workstations," *Int. J. Prod. Res.*, vol. 40, no. 17, pp. 4563–4582, Nov. 2002.

[24] 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, 2002.

[25] G. Y. Liu, P. Li, Z. W. Li, and N. Q. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 7, pp. 1371–1385, 2018.

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

[27] G. Y. Liu, Z. W. Li, K. Barkaoui, and A. M. Al-Ahmari, “Robustness of deadlock control for a class of Petri nets with unreliable resources,” *Inform. Sciences*, vol. 235, no. 6, pp. 259–279, 2013.

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

[29] J. C. Luo, K. Y. Xing, and M. C. Zhou, “Deadlock and blockage control of automated manufacturing systems with an unreliable resource,” *Asian J. Control*, vol. 21, no. 6, pp. 1–12, Nov. 2018.

[30] J. C. Luo, Z. Q. Liu, M. C. Zhou, K. Y. Xing, X. N. Wang, X. L. Li, and H. X. Liu, “Robust deadlock control of automated manufacturing systems with multiple unreliable resources,” *Inform. Sciences*, vol. 479, pp. 401–415, Apr. 2019.

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

[32] 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,” *Inform. Sciences*, vol. 345, pp. 243–256, Jun. 2016.

[33] Y. C. Wu, K. Y. Xing, J. C. Luo, and Y. X. Feng, “Robust deadlock control for automated manufacturing systems with an unreliable resource,” *Inform. Sciences*, vol. 346, pp. 17–28, Jun. 2016.

[34] H. Yue, K. Y. 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.

[35] H. Yue, K. Y. Xing, H. S. Hu, W. M. Wu, and H. Y. Su, “Resource failure and buffer space allocation control for automated manufacturing systems,” *Inform. Sciences*, vol. 450, pp. 392–408, Jun. 2018.

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

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

[38] F. J. Yang, N. Q. Wu, Y. Qao, M. C. Zhou, and Z. W. Li, “Scheduling of single arm cluster tools for anatomic layer deposition process with residency time constraints,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 502–516, Mar. 2017.

[39] X. Lu, M. C. Zhou, A. C. Ammari, and J. Ji, “Hybrid Petri nets for modeling and analysis of microgrid systems,” *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 4, pp. 347–354, Oct. 2016.

[40] J. C. Luo, Z. Q. Liu, M. C. Zhou, and K. Y. Xing, “Deadlock-free scheduling of flexible assembly systems based on Petri nets and local search,” *IEEE Trans. Syst., Man, Cybern., Syst*, 2018.

[41] T. Murata, “Petri nets: properties, analysis and applications,” *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

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

[43] N. Ran, H. Su, and S. Wang, “An improved approach to test diagnosability of bounded Petri nets,” *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 2, pp. 297–303, 2017.

[44] N. Q. Wu and M. C. Zhou, “Modeling, analysis and control of dual arm cluster tools with residency time constraint and activity time variation based on Petri nets,” *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 446–454, Apr. 2012.

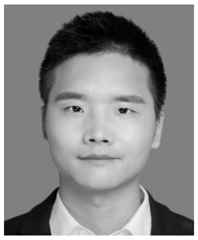
[45] N. Q. Wu, M. C. Zhou, and Z. W. Li, “Short term scheduling of crude oil operations: Petri net based control theoretic approach,” *IEEE Trans. Robot. Autom. Mag.*, vol. 22, no. 2, pp. 64–76, Jun. 2015.

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

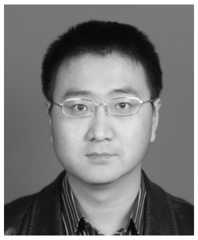
[47] S. W. Zhang, N. Q. Wu, Z. W. Li, T. Qu, and C. D. Li, “Petri net based approach to short term scheduling of crude oil operations with less tank requirement,” *Inform. Sciences*, vol. 417, pp. 247–261, Nov. 2017.

[48] M. C. Zhou and M. P. Fanti, *Deadlock Resolution in Computer-Integrated System*, New York, NY, USA: Marcel Dekker, 2005.

[49] Q. H. Zhu, M. C. Zhou, Y. Qiao, and N. Q. Wu, “Petri net modeling and scheduling of a close down process for time-constrained single-arm cluster tools,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 3, pp. 389–400, Mar. 2018.



Jianchao Luo received the Ph.D. degree in control science & engineering at Xi’an Jiaotong University, in 2016. He joined the Northwestern Polytechnical University, in 2017, where he is currently an Assistant Professor of software engineering. His research interests include deadlock-free scheduling and control of AMSs.



Zhiqiang Liu received the Ph.D. degree in computer science and engineering from Northwestern Polytechnical University, in 2007, where he is currently an Associate Professor of software engineering. His research interests include safety critical software, data analysis, deadlock-free scheduling, and control of AMSs.

Shuogang Wang received the B.S. degree in software engineering from Northwestern Polytechnical University in 2015, where he is pursuing the M.S. degree in software engineering. His research interests include mobile application development and deadlock control of AMSs.

Keyi Xing received the Ph.D. degree in systems engineering from Xi’an Jiaotong University, in 1994. Currently, he is a Professor of systems engineering with the State Key Laboratory for Manufacturing Systems Engineering and the Systems Engineering Institute, Xi’an Jiaotong University. His research interests include control and scheduling of AMSs.