A Petri Net-based Deadlock Avoidance Policy for Flexible Manufacturing Systems with Assembly Operations and Multiple Resource Acquisition

JianChao Luo, ZhiQiang Liu, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Efficient deadlock control policies are very important in the operation of flexible manufacturing systems (FMS). This work focuses on deadlock control problems for a general class of FMS. They have three interesting characteristics from the application point of view. First, flexible routes of parts and assembly operations are allowed. Second, the number of parts of the same type in a product may be more than one. Third, an operation may require multiple resources. To characterize such FMS, a Petri net model that can well deal with all FMS characteristics is developed. Based on it, this work proposes a Banker's algorithm-like deadlock avoidance policy. The proposed policy is proved to be polynomial in the model size. Moreover, experimental results indicate its effectiveness and superiority over the state-of-the-art policies.**

***Index Terms*—Flexible manufacturing systems, deadlock avoidance policy, Petri net, assembly operation, flexible routes.**

I. Introduction

Deadlocks may occur in an automated manufacturing system (AMS) and may cause the system indefinitely blocked. Thus, extensive researchers have devoted to resolving the deadlock problems in AMSs and many deadlock control policies have been proposed. The existing policies mainly include two kinds: prevention and avoidance. The former establishes in advance offline control policies such that the resulting manufacturing operation is deadlock-free [1], [3], [8]-[10], [12]-[15], [19], [23], [27], [28], [32], while the latter consists of online control policies that use feedback information about the current state to keep AMS away from deadlocks [2], [6], [7], [16]-[18], [22], [24], [29], [31]. Most of them do not consider the assembly operation which puts together multiple parts to produce a product.

Recently, the deadlock problem in AMSs with assembly operations has received some research attention. In these systems, deadlocks may result from not only the circuit wait of resources but also the parts waiting for the assembly with other parts. Thus, their deadlock control problem is more difficult than that in AMSs without assembly operations. Research on AMSs with assembly operation can be found in [8], [11]-[13], [24], [29]. Roszkowska [24] performs the deadlock control of AMSs with fork/join material flow, and proves that the problem of finding the maximum permissive deadlock avoidance policy for such systems is NP-hard. Then, she proposes a sub-optimal yet computationally acceptable deadlock avoidance policy to reach a compromise. Fanti *et al.* [8] also investigate such AMS, each of whose processing steps is limited to acquire a single resource only. Based on the discrete event system model, they propose a deadlock avoidance policy by inhibiting or enabling the events involving resource allocation. Wu *et al.* [29] study the same kind of AMSs as in [24] based on a resource-oriented Petri net (PN). A deadlock avoidance policy, which is proved to be computationally efficient and less conservative than the one in [24], is proposed. Hu and Zhou [13] examine AMSs with assembly operations and multiple resource acquisition but without flexible routes based on their PN models. They simplify the control problem by restricting the system behavior by using a special assumption. Then, they establish the equivalence relationship between the liveness and deadlock-freeness of the restricted system. At last, they develop a mathematical programming method to identify deadlocks and remove them iteratively. Hu *et al.* [12] further deal with a general class of AMSs with flexible routes, assembly operations and multiple resource acquisition. They propose a novel class of systems and their corresponding Petri net models, which can well deal with all features. Then, they establish the relationship between the nonliveness and the absence of undermarked siphons of their net models. Finally, they give an algebraic method such that the siphons can be derived and controlled in an iterative way. A limitation of their work is that the number of parts of the same type in a product is limited by one.

We pay attention to AMSs in [12]. Differing from [12], a product in our system can be made of multiple types of parts and the number of parts of the same type may be more than one. Clearly, our studied AMS is more general than that in [12]. To characterize it, we have to develop a Petri net model that can well deal with all AMS characteristics. Based on it, we propose

Corresponding authors: M. C. Zhou and J. C. Luo.

J. C. Luo and Z. Q. Liu are with the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China (e-mails: luojianchao@nwpu.edu.cn, zqliu@nwpu.edu.cn).

M. C. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

a Banker's algorithm-like deadlock avoidance policy. It is proved to be polynomial with respect to the net size and tested to be more permissive than existing ones.

This paper is structured as follows. Section II reviews the definitions of PN and develops the PN model of the studied AMSs. Based on it, a deadlock avoidance policy is proposed in Section III. Section IV demonstrates its effectiveness. We conclude this paper in Section V.

II. Basic PN Definitions and AMSs PN Models

First, we recall the basics of PNs. Their more details can be found in [21] and recent applications in [20], [25], [30], [33]. Next, we describe the studied AMSs and develop their PN models.

A. Basic Definitions of PNs

Let $Z = \{0, 1, 2, \ldots\}$, $Z^+ = \{1, 2, 3, \ldots\}$, and $Z_k = \{1, 2, \ldots, k\}$ where $k \in Z^+$. A PN is a 4-tuple $N = (P, T, F, W)$, where P is a set of places, T is a set of transitions with $P \cap T = \varnothing$, $P \neq \varnothing$, and $T \neq \varnothing$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, and W: $F \to Z^+$ is a mapping that assigns to each arc a positive integer or weight.

The preset and postset of a vertex $x \in P \cup T$ are defined as ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$, respectively. Given $X \subseteq P \cup T$, we define ${}^\bullet X = \cup_{x \in X} {}^\bullet x$ and $X^\bullet = \cup_{x \in X} x^\bullet$. A path in N is a sequence of vertices $\alpha = x_1 x_2 \ldots x_k$, where $x_i \in P \cup T$ and $(x_i, x_{i+1}) \in F$ $\forall i \in Z_{k-1}$.

A marking of N is a mapping M: $P \to Z$. Given a place $p \in P$ and a marking M, $M(p)$ denotes the number of tokens in p at M. (N, M_0) is a marked PN with an initial marking M_0.

A transition $t \in T$ is enabled to fire at M, denoted by $M[t\rangle$, if $\forall p \in {}^\bullet t$, $M(p) > W(p, t)$. Firing t at M generates a new marking M', denoted as $M[t\rangle M'$, where $M'(p) = M(p) - W(p, t) + W(t, p)$. A sequence of transitions $\alpha = t_1 t_2 \ldots t_k$ is feasible from M if there exists $M_i[t_i\rangle M_{i+1}$, $i \in Z_k$, where $M_1 = M$. We call that M_{i+1} is reachable from M. The set of all markings reachable from M is denoted by $R(N, M)$.

B. Petri Net Modeling of AMSs

An AMS studied in this paper consists of m types of resources and can manufacture and assemble l types of products. Let $R = \{r_i \mid i \in Z_m\}$ be the set of resource types. Each type of resources may be a kind of machines, buffers or robots. The capacity of r_i is denoted as C_i, where $C_i \in Z^+$. Let n be the number of part types, $J = \{J_i \mid i \in Z_n\}$ be the set of part types, and Ψ_i be the number of type-i parts to be processed. The same type of raw parts can only be used to assemble a type of products.

A processing route of a part is a sequence of manufacturing and assembly operations or activities. In a manufacturing activity, parts are only processed, while in an assembly activity, two or more parts are assembled into a (final or intermediate) product. A part may have more than one processing route and can select its route during its processing. Let $\Theta = \{\theta_i \mid i \in Z_{|\Theta|}\}$ be the set of all processing routes in the system, and $\Theta(i) \subseteq \Theta$ denote the set of routes of J_i, where $i \in Z_n$. $\forall i, j \in Z_n$ and $i \neq j$, $\Theta(i) \cap \Theta(j) = \varnothing$, $\Theta(i) \neq \varnothing$, and $\Theta(j) \neq \varnothing$. Route θ_i can be expressed as $\theta_i = o_{is} o_{i1} o_{i2} \ldots o_{iL_i} o_{ie}$, where o_{ij} is the jth activity, L_i is the total number of activities, and o_{is} (o_{ie}) denotes the start (end) activity in θ_i.

In our PN model, route θ_i is modeled by a path $\alpha_i = p_{is} t_{i1} p_{i1} t_{i2} p_{i2} t_{i3} \ldots p_{iL_i} t_{i(L_i+1)} p_{ie}$, where p_{is} (p_{ie}) represents the activity o_{is} (o_{ie}), p_{ij} is an activity place representing activity o_{ij}, and t_{ij} ($t_{i(j+1)}$) represents the start (completion) of o_{ij}. We call α_i as an *activity path* as all places in it are activity ones. Hence, the marked PN model of θ_i can be denoted as:

$$(N_i, M_{i0}) = (P_i \cup \{p_{is}, p_{ie}\}, T_i, F_i, W_i, M_{i0}),\ i \in Z_{|\Theta|}$$

where $P_i = \{p_{i1}, \ldots, p_{iL_i}\}$, $T_i = \{t_{i1}, t_{i2}, \ldots, t_{i(L_i+1)}\}$, and $F_i = \{(p_{is}, t_{i1}), (t_{i1}, p_{i1}), \ldots, (p_{iL_i}, t_{i(L_i+1)}), (t_{i(L_i+1)}, p_{ie})\}$, and $\forall (p, t) \in F_i$, $W_i(p, t)$ is the number of parts required at p to fire t. M_{i0} is the initial marking, $M_{i0}(p) = 0$, $\forall p \in P_i \cup \{p_{ie}\}$, and if $\theta_i \in \Theta(k)$, then $M_{i0}(p_{is}) = \Psi_k$.

Note that $\exists (p, t) \in F_i \ni W_i(p, t) > 1$. That means multiple raw parts of the same type may be assembled into a single product. Suppose that the numbers of raw parts of the same type needed by different routes to assemble a product are equal. If $\theta_i \in \Theta(k)$, then the number of raw type-k parts needed to assemble a product is $f_k = W_i(p_{is}, t_{i1}) \times W_i(p_{i1}, t_{i2}) \times \ldots \times W_i(p_{iL_i}, t_{i(L_i+1)})$.

Different processing routes may share some identical activities. For simplicity, identical activities are merged as one, i.e., they have the same activity place and start transition. They may have different completion transitions. In this situation, the activity place p is called a split place since $|p^\bullet| > 1$. From a split place, a part can select its processing route. For any activity place p, suppose that $W(p, t_1) = W(p, t_2)$ $\forall t_1, t_2 \in p^\bullet$. Then, let $W(p)$ denote the weight of the arc from p to any one of the transitions in $p^\bullet$ for simplicity.

The same type of raw parts can only be used to assemble a type of products, and thus there is a single start and end activity place for a type of parts. The set of all initial activity places is denoted as $\{p_{is} \mid i \in Z_n\}$. The number of end activity places is equal to the number of product types, i.e., l, and the set of end activity places is denoted as $\{p_{ie} \mid i \in Z_l\}$. Since there are assembly activities, an end activity place may be shared by multiple types of raw parts. Given an end activity p_{ie}, let $\xi(p_{ie})$ be the set of part types that share p_{ie}.

To each resource type $r_i \in R$, we assign a place, called a resource place and denoted also by r_i for simplicity. Tokens in r_i indicate the number of available type-i resources. The initial marking of r_i is C_i. Let P_R denote the set of all resource places.

In this paper, each activity may require multiple types of resources. The resource requirement of activity place p is denoted as a vector $R(p) = (\gamma_1(p), \ldots, \gamma_m(p))^T$, where $\forall i \in Z_m$, $\gamma_i(p)$ is the number of type-i resources required by an activity in p. For economy of space, $\sum_{i \in Z_m} \gamma_i(p) r_i$ is used to denote $R(p)$. Then $\forall i \in Z_m$ and $\gamma_i(p) > 0$, add arcs with weight $\gamma_i(p)$ from r_i to each transition in ${}^\bullet p$, denoting the allocation of r_i, and arcs with weight $\gamma_i(p) \times W(p)$ from each transition in $p^\bullet$ to r_i, denoting the release of r_i. Let F_R denote the set of arcs related with resource places, and W_R denote its corresponding weight mapping. The whole system can be modeled by the following marked PN:

$$(N, M_0) = (P \cup P_s \cup P_e \cup P_R, T, F, W, M_0)$$

where $P = \{P_i \mid i \in Z_{|\Theta|}\}$, $P_s = \{p_{is} \mid i \in Z_n\}$, $P_e = \{p_{ie} \mid i \in Z_l\}$, $T = \{T_i \mid i \in Z_{|\Theta|}\}$, $F = F_\Theta \cup F_R$, $F_\Theta = \{F_i \mid i \in Z_{|\Theta|}\}$, and $\forall (p, t) \in F_i$, $W(p, t) = W_i(p, t)$, $\forall (p, t) \in F_R$, $W(p, t) = W_R(p, t)$. The initial marking M_0 is defined as $M_0(p_{is}) = \Psi_i$, $\forall p_{is} \in P_s$; $M_0(p) = 0$, $\forall p \in P \cup P_e$; and $M_0(r_i) = C_i$, $\forall r_i \in P_R$.

In (N, M_0), if $t \in T$, $|{}^\bullet t| > 1$, t is called an assembly transition, $p \in {}^\bullet t$ is called an assembly activity place, and if $|t^\bullet| > 1$, t is called a disassembly transition. In the following, we refer to the above PN model as an *extended assembly PN* (EAPN).

This work considers only initial markings that represent no activity in $P \cup P_e$, allow the completion of each product in isolation, and ensure all raw parts to be manufactured or assembled into products. They are called *acceptable* initial markings, as formally defined next.

Definition 1: Given an EAPN $N = (P \cup P_s \cup P_e \cup P_R, T, F, W)$, M_0 is acceptable for N if: 1) $M_0(p) = 0$, $\forall p \in P \cup P_e$; 2) $M_0(p_{is}) > 0$ and $M_0(p_{is}) \,\%\, f_i = 0$, $\forall i \in Z_n$; 3) $M_0(r_j) \geq \gamma_j(p) \times W(p)$, $\forall p \in P$, $\forall j \in Z_m$; 4) $\forall t \in T$ if $|t^\bullet \cap P| \geq 2$, $M_0(r_j) \geq \sum_{p \in P \cap t^\bullet}(\gamma_j(p) \times W(p))$, $\forall j \in Z_m$; 5) $\forall t \in T$ if $|{}^\bullet t \cap P| \geq 2$, $M_0(r_j) \geq \sum_{p \in P \cap {}^\bullet t}(\gamma_j(p) \times W(p))$, $\forall j \in Z_m$; and 6) $M_0(p_{us})/f_u = M_0(p_{vs})/f_v$, $\forall u, v \in \xi(p_{ie})$, $\forall p_{ie} \in P_e$.

An acceptable M_0 is the precondition to study a deadlock problem in EAPN. If this condition is not satisfied, the production of certain products is doomed to failure owing to the insufficiency of resources, or some type of raw parts is doomed to surplus owing to the insufficiency of other type of parts. Deadlock-free control under such defect is impossible and meaningless in practice. Thus, we suppose that M_0 is always acceptable in the following discussions.

Starting from M_0, when all activities of all parts are finished, N reaches a final marking, denoted as M_f, which is defined as follows. $\forall p_{ie} \in P_e$, $M_f(p_{ie}) = M_0(p_{js})/f_j$, where $j \in \xi(p_{ie})$; $\forall p \in P \cup P_s$, $M_f(p) = 0$; and $\forall r_i \in P_R$, $M_f(r_i) = C_i$. For a marking $M \in R(N, M_0)$, M is *safe* if $M_f \in R(N, M)$; and otherwise, M is *unsafe*.

Example 1: Consider an AMS with six types of resources r_1-r_6, i.e., $R = \{r_i \mid i \in Z_6\}$. $C_1 = C_2 = C_4 = 2$, and $C_3 = C_5 = C_6 = C_7 = 1$. There are four types of raw parts, which can be manufactured and assembled into two types of products. The part type set is $J = \{J_i \mid i \in Z_4\}$. Type-1 parts are first manufactured on r_1, and then manufactured on r_2 and r_3 at the same time. Type-2 parts are manufactured on r_2 and r_1 sequentially, and type-3 parts are manufactured on r_4 and r_5 sequentially. Then type-2 and type-3 parts are assembled on r_3 into intermediate products. Type-4 parts are manufactured on r_4 and r_5 (or r_6 and r_5) sequentially. Then they are assembled with the intermediate products on r_7 into final products. Each of types-1~3 parts has only one processing route, i.e., $\theta_1 = o_{1s}o_{11}o_{12}o_{1e}$, $\theta_2 = o_{2s}o_{21}o_{22}o_{23}o_{24}o_{2e}$, and $\theta_3 = o_{3s}o_{31}o_{32}o_{33}o_{34}o_{3e}$. Type-4 parts have two processing routes denoted as $\theta_4 = o_{4s}o_{41}o_{42}o_{43}o_{4e}$, and $\theta_5 = o_{5s}o_{51}o_{52}o_{53}o_{5e}$. θ_1-θ_5 are modeled by $p_{1s}t_{11}p_{11}t_{12}p_{12}t_{13}p_{1e}$, $p_{2s}t_{21}p_{21}t_{22}p_{22}t_{23}p_{23}t_{24}p_{24}t_{25}p_{2e}$, $p_{3s}t_{31}p_{31}t_{32}p_{32}t_{23}p_{23}t_{24}p_{24}t_{25}p_{2e}$, $p_{4s}t_{41}p_{41}t_{42}p_{42}t_{24}p_{24}t_{25}p_{2e}$, and $p_{4s}t_{51}p_{51}t_{52}p_{42}t_{24}p_{24}t_{25}p_{2e}$, respectively. The weights of the arcs in F_Θ are all 1 except that $W(p_{22}, t_{23}) = 2$. Since o_{23} and o_{33} represent the same activity, their activity places and start transitions are the same. So are the activity places and start transitions of o_{42} and o_{52}; o_{24}, o_{34}, o_{43} and o_{53}; and o_{2e}, o_{3e}, o_{4e} and o_{5e}. p_{4s} is a split place since $|p^\bullet_{4s}| > 1$. $R(p_{11}) = r_1$, $R(p_{12}) = 2r_2 + r_3$, $R(p_{21}) = 2r_2$, $R(p_{22}) = r_1$, $R(p_{23}) = r_3$, $R(p_{24}) = r_7$, $R(p_{31}) = 2r_4$, $R(p_{32}) = r_5$, $R(p_{41}) = 2r_4$, $R(p_{42}) = r_5$, and $R(p_{51}) = r_6$. The EAPN model of the whole system is shown in Fig. 1.

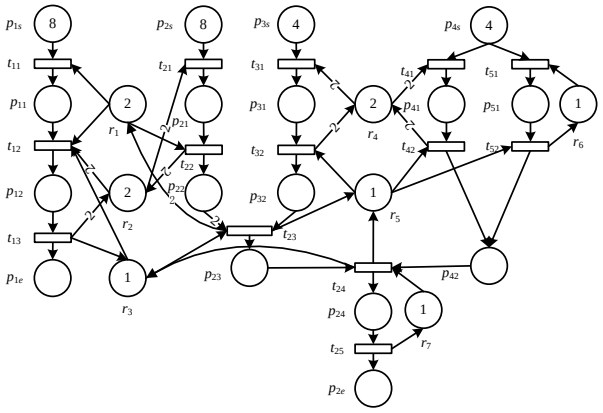

Fig. 1. EAPN of the AMS in Example 1.

Statement 1: EAPN contains AEMG [12], S^*PR [6], S^4R [27] (or S^3PGR [22]), AMG [4], S^3PR [14], LS^3PR [7], and PPN [2], [26], [31].

Proof: Both EAPN and AEMG can deal with flexible routings, assembly operations, and multiple resource acquisition. EAPN considers the situation that $\exists (p, t) \in F_\Theta \ni W(p, t) > 1$, while such a situation is not considered in AEMG. Thus, EAPN contains AEMG. From [12], we know that AEMG contains S^4R (or S^3PGR), AMG, S^3PR, LS^3PR, and PPN. Thus, EAPN contains them, too. S^*PR can deal with flexible routings and multiple resource acquisition, but cannot deal with assembly operations. However, EAPN cannot deal with them all. Thus, EAPN contains S^*PR. ■

III. Modified Banker's Algorithm for Deadlock Avoidance in EAPN

A deadlock avoidance policy (DAP) is to restrict a system to safe reachable markings by disabling an appropriate set of enabled transitions such that the unsafe markings are not reachable from the initial one. Different Banker's algorithm-like DAPs [6], [16]-[18] have been proposed for AMSs without assembly operations. They all verify the safeness of a given marking by the following procedure: first, find an activity able to be terminated using the resources it holds plus the available resources; if no such activity exists, the marking is considered to be unsafe; otherwise, add the resources used by it to the free resources, withdraw it from the set of active activities, and iterate the procedure. If every activity can be removed from the set of active activities, the marking is safe. In AMSs with assembly operations, to terminate an activity needs not only the free resources but also the completion of other activities. Thus, DAPs in [6], [16]-[18], cannot be used to deal with EAPN.

To address the deadlock avoidance problem of EAPN, a modified Banker's algorithm (MBA) is proposed. It contains

two parts, i.e., MBA_1 and MBA_2. MBA_1 tries to terminate all activities, and MBA_2 tries to move a token that corresponds to an activity to its first-met assembly activity place. They are executed iteratively. In the following, MBA_1, MBA_2, and MBA are proposed sequentially.

A. MBA_1

When an activity is terminated, the resources occupied by it are freed. Then, they can be used to manufacture or assemble other parts. Thus we try to terminate such activities first. In order to explain it clearly, let us introduce some new notations.

An activity path is called an *ending activity path* if the last activity place is an end one. Let $p \in P \cup P_s$ be an activity place, M be a marking, and A(p) denote the set of all ending activity paths starting from p. Given an activity path α, let $\tau(\alpha)$ be the transition sequence in α. An activity path α is called *executable* at M, if $M[\tau(\alpha)>$. A token in an activity place is corresponding to an activity. A token or activity in p is called *terminable* at M if there is an executable ending activity path in A(p).

To detect the safeness of a marking M, we need to terminate all activities in P, and only part of activities in P_s. Given $M \in R(N, M_0)$, let $\vartheta(M)$ be a making reduced from M, which is defined as follows. $\forall p \in P \cup P_R$, $\vartheta(M)(p) = M(p)$; $\forall p_{is} \in P_s$, $\vartheta(M)(p_{is}) = M(p_{is}) - \min\{\lfloor M(p_{js})/f_j \rfloor \mid j \in \xi(p_{ke})\} \times f_i$, where p_{ke} is the end activity place of p_{is}; and $\forall p_{ie} \in P_e$, $\vartheta(M)(p_{ie}) = M(p_{ie}) + \min\{\lfloor M(p_{js})/f_j \rfloor \mid j \in \xi(p_{ie})\}$. If all activities at $\vartheta(M)$ can be terminated from M, then only some activities are remaining in P_s. These activities can be terminated since M_0 is acceptable. Thus, $\vartheta(M)$ has the same safeness as M does. It means that we can know the safeness of M by detecting the safeness of $\vartheta(M)$. Since $\vartheta(M)$ has a smaller number of activities needed to be terminated than M, detecting its safeness is easier.

Lemma 1: Given $M \in R(N, M_0)$, let $M_1 = \vartheta(M)$ be a making reduced from M. The number of activities in P at M_1 is not greater than $\mathcal{C} = \sum_{i \in Z_m} C_i$. The activities at M_1 can lead to at most $\mathcal{C}$ products.

Proof: Each activity in P occupies at least one resource instance. Thus, the maximum number of activities in P at M_1 is bounded by the total number of resource instances, i.e., $\mathcal{C} = \sum_{i \in Z_m} C_i$. At M_1, a product may be manufactured and assembled in three ways: it is manufactured from a single activity in P; it is assembled from multiple activities in P; and it is assembled from one or more activities in P and one or more activities P_s. It means that each product manufactured and assembled at M_1 needs at least one activity in P. Thus, the activities at M_1 can lead to at most $\mathcal{C}$ products. ■

Example 2: Consider EAPN (N, M_0) in Fig. 1. Let $M = 7p_{1s} + p_{11} + 8p_{2s} + 4p_{3s} + 4p_{4s} + r_1 + 2r_2 + r_3 + 2r_4 + r_5 + r_6 + r_7 \in R(N, M_0)$. A($p_{11}$) = {$\alpha$} = {$p_{11}t_{12}p_{12}t_{13}p_{1e}$} and $\tau(\alpha) = t_{12}t_{13}$. Since $M[t_{12}t_{13}>$, α is executable at M, and the activity in p_{11} is terminable at M. $\vartheta(M) = p_{11} + 7p_{1e} + 4p_{2e} + r_1 + 2r_2 + r_3 + 2r_4 + r_5 + r_6 + r_7$. Only one activity, i.e., the activity in p_{11}, needs to be terminated at $\vartheta(M)$. After its termination at $\vartheta(M)$, M_f is reached. Thus, $\vartheta(M)$ is safe. After terminating the activity in p_{11} at M, the system reaches a new marking $M_1 = 7p_{1s} + p_{1e} + 8p_{2s} + 4p_{3s} + 4p_{4s} + 2r_1 + 2r_2 + r_3 + 2r_4 + r_5 + r_6 + r_7$. Since M_0 is acceptable, all activities at M_1 can be terminated, i.e., M_1 is safe. Thus, M is safe, and $\vartheta(M)$ has the same safeness as M does.

Now MBA_1 is presented as follows.

MBA_1

Input: a marking $M \in R(N, M_0)$;

Output: a marking M_1; /* the resulting marking after terminating some activities from M */

```
Begin
initialize: M1 = ϑ(M);
while (true) {
   flag = false;
   for (p ∈ P ∪ Ps) {
      if (M1(p) > 0) {
         if (∃α ∈ A(p) ∋ M1[τ(α)>) {/* find an executable ending activity path */
            let M1[τ(α)> M2 and M1 = M2;
            flag = true; break; }
      }/* end for if (M1(p) > 0) */
   }/* end for (p ∈ P ∪ Ps) */
   if (!flag) return M1; /* no activity is terminable */
}/* end while (true) */
End
```

Note that when some terminable activity is found, MBA_1 restarts the searching procedure from the beginning. This is because when some activity is terminated, some resources are freed, and they may make some activities terminable. Such a restarting operation ensures that no more activity is terminable after the execution of MBA_1.

Theorem 1: MBA_1 is of polynomial complexity.

Proof: The cost of finding an executable ending activity path from an activity place p is $O(|\Theta| \times |T|)$ because there are at most $|\Theta|$ ending activity paths starting from p, and the length of the transition sequence in an activity path is no longer than $|T|$. Thus, the cost of the for-loop is $O(|P \cup P_s| \times |\Theta| \times |T|)$. By Lemma 1, the activities at a reduced marking can lead to at most $\mathcal{C}$ products. Thus, the while-loop can execute at most ($\mathcal{C}$ + 1) times, and $O(MBA_1) = O(|P \cup P_s| \times |\Theta| \times |T| \times \mathcal{C})$. Mostly, $|P \cup P_s|$ and $|T|$ are nearly the same and the number of ending activity paths starting from an activity place is one. Thus, $O(MBA_1)$ can be roughly evaluated as $O(\mathcal{C} \times |T|^2)$. ■

B. MBA_2

There are two cases where an activity is not terminable: a) the resources needed to terminate it are occupied by other activities and b) the activities needed to terminate it are not in the assembly activity place. To make some activity terminable, we design MBA_2 to move a token which corresponds to an activity to a first-met assembly activity place such that its current occupied resources can be freed and the activities waiting for it to execute the assembly operation can proceed. In order to explain this part, we need the following new notations.

An activity path $\alpha = p_0t_1p_1t_2 \ldots p_it_{(i+1)}p_{(i+1)}$ is called an *assembly activity path* if p_1, …., and p_i are all non-assembly activity places but $p_{(i+1)}$ is. Let $p \in P \cup P_s$ be an activity place

and $\underline{A}(p)$ denote the set of all assembly activity paths starting from p. An activity or token in p is called *mountable* at a marking if there is an executable assembly activity path in $\underline{A}(p)$.

Given an EAPN (N, M_0), let $\Xi(N)$ denote a sequence of all the activity places in $P \cup P_s$ from which there is an activity path to an assembly activity place. Given a marking $M \in R(N, M_0)$, let $\chi(M)$ = true or false denote if there is a mountable activity at M or not. Now MBA_2 is presented as follows.

MBA_2
Input: a marking $M \in R(N, M_0)$;
Output: M_1 and $\chi(M)$; /*M_1 is the result marking after moving a token to an assembly activity place */
Begin
initialize: $M_1 = M$; $\chi(M)$ = false;
for ($p \in \Xi(N)$) {
  if ($M_1(p) > 0$) {
    if ($\exists \alpha \in \underline{A}(p)$ э $M_1[\tau(\alpha)>$) {/* find an executable assembly activity path */
      let $M_1[\tau(\alpha)> M_2$ and $M_1 = M_2$;
      $\chi(M)$ = true; break;}
  } /* end for if ($M_1(p) > 0$) */
} /* end for ($p \in \Xi(N)$) */
return M_1 and $\chi(M)$;
End

Theorem 2: MBA_2 is of polynomial complexity.

Proof: The cost of finding an executable assembly activity path from an activity place p is $O(|\Theta| \times |T|)$ because there are at most $|\Theta|$ assembly activity paths starting from p, and the length of the transition sequence in an assembly activity path is no longer than $|T|$. Thus, $O(MBA_2) = O(|\Xi(N)| \times |\Theta| \times |T|) = O(|P \cup P_s| \times |\Theta| \times |T|)$. Mostly, $|P \cup P_s|$ and $|T|$ are nearly the same and the number of assembly activity paths starting from an activity place is one. Thus, $O(MBA_2)$ can be roughly evaluated as $O(|T|^2)$. ■

C. MBA

MBA iteratively executes MBA_1 and MBA_2 until the final marking is reached or no activity is terminable and no activity is mountable. Given a marking $M \in R(N, M_0)$, let $\delta(M)$ = true or false denote if MBA accepts it or not. Now MBA is presented as follows.

MBA
Input: a marking $M \in R(N, M_0)$;
Output: $\delta(M)$; /* $\delta(M)$ = true or false */
Begin
initialize: $M_1 = M$; $\delta(M)$ = false; flag = true;
while (flag) {
  $M_2 = MBA_1(M_1)$;
  if ($M_2 = M_f$) { $\delta(M)$ = true; break; }
  (M_1, flag) = $MBA_2(M_2)$;
}
return $\delta(M)$;
End

Theorem 3: MBA is of polynomial complexity.

Proof: By Lemma 1, we know that the activities at M_1 can lead to at most $\mathbb{C}$ products. Each product needs at most $|T|$ assembly operations. Thus, the while-loop can execute at most $\mathbb{C} \times |T|$ times, and $O(MBA) = O(\mathbb{C} \times |T| \times (O(MBA_1) + O(MBA_2)))$. Both MBA_1 and MBA_2 are of polynomial complexity, so is MBA. Mostly, the number of assembly operations needed by a product is one. Thus, $O(MBA)$ can be roughly evaluated as $O(\mathbb{C}^2 \times |T|^2)$. ■

Theorem 4: MBA is a DAP for EAPN.

Proof: Suppose that there is an unsafe marking $M \in R(N, M_0)$ such that $\delta(M)$ = true. Since $\delta(M)$ = true, there is a sequence of transitions τ э $M[\tau> M_f$. It is a contradiction with the assumption that M is unsafe. Thus, all markings accepted by MBA are safe. ■

Although MBA is not optimal, its improvement over the existing work is prominent. The original Banker's algorithm [5] and existing Banker's algorithm-like DAPs [6], [16]-[18] can only deal with systems where parts are independent. However, MBA can deal with systems where parts are independent or dependent. Such an improvement makes our algorithm capable of handling more general systems than those in the existing literatures. Besides, MBA together with some rules to be presented in the next section makes a DAP that admits much more markings than existing deadlock controllers for AMSs with both assembly operations and multiple resource acquisition, to be shown later.

D. Improvement

Since a token in an assembly activity place still occupies resources, the order of moving tokens to their first-met assembly activity places may affect the performance of MBA. Some examples are taken to illustrate such phenomenon, and then three related rules are proposed.

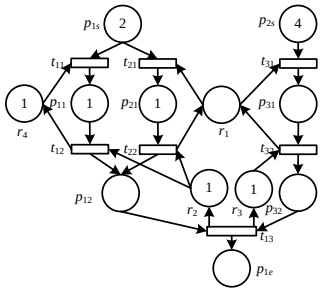

Fig. 2. An illustrative EAPN for Example 3.

Example 3: Consider EAPN in Fig. 2. At marking $M = 2p_{1s} + p_{11} + p_{21} + 4p_{2s} + r_2 + r_3 + r_4$, no activities are terminable. Then, MBA_2 tries to move a token to an assembly activity place. At M, the tokens in p_{1s}, p_{11}, and p_{21} are all mountable. If MBA_2 first move a token in p_{1s} to p_{12}, the system reaches $M_1 = p_{1s} + p_{11} + p_{12} + p_{21} + 4p_{2s} + r_3 + r_4$. No activity is terminable or mountable at M_1, and M_1 is an unsafe marking. If MBA_2 first move the token in p_{11} to p_{12}, the system reaches $M_2 = 2p_{1s} + p_{12} + p_{21} + 4p_{2s} + r_3 + 2r_4$. No activity is terminable or mountable at M_2, and M_2 is an unsafe marking. However, if MBA_2 first moves the token in p_{21} to p_{12}, the system reaches $M_3 = 2p_{1s} + p_{11} + p_{12} + 4p_{2s} + r_1 + r_3 + r_4$. At M_3, an activity in p_{2s} is terminable. After it

terminates via MBA_1, the system reaches $M_4 = 2p_{1s} + p_{11} + 3p_{2s} + r_1 + r_2 + r_3 + r_4$. Obviously, M_4 is safe. Thus, performing activities in p_{21} first is better than doing so in p_{11} or p_{1s} first.

Tokens in the start activity places do not occupy any resources. Yet moving them to assembly activity places must occupy some resources that may be required by other activities, and may thus cause a deadlock. As shown in Example 3, moving a token in p_{1s} to p_{12} at M causes such deadlock. Thus, we move them in the end, which is ensured by the following rule.

Rule 1: Move the tokens in p_{js} ($j \in Z_n$) after those in P.

If all occupied resources by a token are not required by any other activities, then moving it to an assembly activity place may occupy some resources that are required by other activities, and hence may cause a deadlock. As shown in Example 3, moving a token in p_{11} to p_{12} at M causes such deadlock. Thus, we move such tokens behind other tokens in P, which is ensured by the following rule.

Rule 2: Let p be an activity place and $\{r_j \mid \exists j \in Z_m \ni \gamma_j(p) > 0\}$ be the set of resources occupied by a token in p. If $\forall r_k \in \{r_j \mid \exists j \in Z_m \ni \gamma_j(p) > 0\}$, $\nexists p_1 \in P\backslash\{p\} \ni \gamma_k(p_1) > 0$, then move the tokens in p after those in $P\backslash\{p\}$.

Example 4: Consider EAPN in Fig. 1. Let $M = 8p_{1e} + 6p_{2s} + 2p_{22} + 4p_{3s} + 4p_{4s} + 2r_2 + r_3 + 2r_4 + r_5 + r_6 + r_7 \in R(N, M_0)$. No activity is terminable at M. Then, MBA_2 tries to move a token to an assembly activity place. At M, the tokens in p_{3s} and p_{4s} are all mountable. If MBA_2 first moves a token in p_{4s} to p_{42}, the system reaches $M_1 = 8p_{1e} + 6p_{2s} + 2p_{22} + 4p_{3s} + 3p_{4s} + p_{42} + 2r_2 + r_3 + 2r_4 + r_6 + r_7$. No activity is terminable or mountable at M_1, and M_1 is an unsafe marking. However, if MBA_2 first moves a token in p_{3s} to p_{32}, the system reaches $M_2 = 8p_{1e} + 6p_{2s} + 2p_{22} + 3p_{3s} + p_{32} + 4p_{4s} + 2r_2 + r_3 + 2r_4 + r_6 + r_7$. At M_2, the token in p_{22} is mountable. After MBA_2 moving it to p_{23}, the system reaches $M_3 = 8p_{1e} + 6p_{2s} + p_{23} + 3p_{3s} + 4p_{4s} + 2r_1 + 2r_2 + 2r_4 + r_5 + r_6 + r_7$. At M_3, an activity in p_{4s} is terminable. After it terminates via MBA_1, the system reaches $M_4 = 8p_{1e} + 6p_{2s} + p_{2e} + 3p_{3s} + 3p_{4s} + 2r_1 + 2r_2 + r_3 + 2r_4 + r_5 + r_6 + r_7$. Since M_0 is acceptable, all activities at M_4 can be terminated, i.e., M_4 is safe. Thus, moving tokens in p_{3s} first is better than doing so in p_{4s} first.

For EAPN in Fig. 1, p_{42} is an assembly place with $\gamma_5(p_{42}) = C_5$, and $\gamma_5(p_{32}) > 0$. The assembly completion of the activity in p_{32} is needed by the activity in p_{42}. If a token is moved to p_{42} and no token is in p_{23} at the resulting marking, no token can be moved to p_{32} from the resulting marking. Hence, it is an unsafe marking. As shown in Example 4, after moving a token in p_{4s} to p_{42} at M, none of the tokens in p_{3s} can be moved to p_{32}. Hence, an unsafe marking, i.e., M_1 is reached. Hence, we need the following rule to avoid such situation.

Rule 3: Let $\alpha_1 = p_{11}t_{12}\ldots p_{1i}t_{1(i+1)}p_{1(i+1)}$ be an assembly activity path, and $\alpha_2 = p_{21}t_{22}\ldots p_{2j}t_{2(j+1)}p_{2(j+1)}$ be an activity path such that $p^{\bullet}_{1(i+1)} = p^{\bullet}_{2(j+1)}$. If $\exists k \in Z_m$ and $q \in Z_j \ni \gamma_k(p_{1(i+1)}) = C_k$ and $\gamma_k(p_{2q}) > 0$, then move the tokens in p_{1u} ($\forall u \in Z_i$) after those in p_{2v} ($\forall v \in Z_q$).

To improve the performance of MBA, we apply Rules 1-3 on $\Xi(N)$ recursively, and finally obtain a new sequence, denoted as $\Xi_3(N)$. Take EAPN in Fig. 1, i.e., (N, M_0), for example. Suppose that $\Xi(N) = \{p_{21}, p_{22}, p_{2s}, p_{51}, p_{31}, p_{32}, p_{3s}, p_{41}, p_{4s}\}$. We first apply Rule 1 on $\Xi(N)$, and obtain $\Xi_1(N) = \{p_{21}, p_{22}, p_{51}, p_{31}, p_{32}, p_{41}, p_{2s}, p_{3s}, p_{4s}\}$. Then, we apply Rule 2 on $\Xi_1(N)$, and obtain $\Xi_2(N) = \{p_{21}, p_{22}, p_{31}, p_{32}, p_{41}, p_{51}, p_{2s}, p_{3s}, p_{4s}\}$. Finally, we apply Rule 3 on $\Xi_2(N)$, and obtain $\Xi_3(N) = \{p_{21}, p_{22}, p_{31}, p_{32}, p_{3s}, p_{41}, p_{51}, p_{2s}, p_{4s}\}$. If $\Xi(N)$ is used in MBA_2, the number of reachable markings of (N, M_0) under MBA is 215. However, if $\Xi_3(N)$ is used in MBA_2, the number of reachable markings of (N, M_0) under MBA is 450. That means different sequences of all activity places in $\Xi(N)$ may lead to different control performance. This inspires another interesting issue of how to obtain the optimal sequence, which is nevertheless out of the scope of this paper. In the following test, all activity places in $\Xi(N)$ are ordered by their indexes at first. Then, we apply Rules 1-3 on the ordered sequence recursively and obtain a new sequence. At last, the newly obtained sequence is used in MBA_2.

IV. Illustrative Examples and Comparative Studies

A. Illustrative Examples

To the best of our knowledge, no existing deadlock controllers can be applied to EAPN directly. To test the effectiveness of MBA, we have to apply it to a subclass of systems. Existing deadlock controllers that have addressed both assembly operations and multiple resource acquisition can be found in [12], [13], [24] and [29]. In the following, we will compare MBA with them by manufacturing systems in [12] and [13]. As shown in [29], their controller is better than that in [24]. Therefore, we compare the proposed one with those in [12], [13], and [29]. For simplicity, let H_{13}, H_{15}, and W_{08} denote them, respectively.

In W_{08}, the resources are divided into two classes: one is for the base components and another is for parts. Suppose that there are u and v kinds of buffers, $B_{1\text{-}u}$ and $b_{1\text{-}v}$, for the base components and parts, respectively. Note that a kind of buffers is corresponding to a type of resources in this paper. A marking M is accepted by W_{08} if it satisfies four conditions, in which two are as follows:

a) At M, at most one kind of buffers in $B_{1\text{-}u}$ is full; and
b) At M, at most one kind of buffers in $b_{1\text{-}v}$ is full.

The first illustrative manufacturing system comes from [12]. Its PN model is denoted as (N_1, M_{10}) and shown in Fig. 3. In N_1, start activity places are the same as end ones. By taking all of them as end activity places, then MBA can be applied to it directly. Since there are flexible routes in N_1, H_{15} cannot be applied to it. For (N_1, M_{10}), the first two conditions in W_{08} can be changed to four constraints as follows:

1) $M(p_{26}) + M(p_{30}) \geq 1$;
2) $M(p_3) + M(p_6) + M(p_7) + M(p_9) + M(p_{10}) + M(p_{11}) + M(p_{13}) \leq 1 \,||\, M(p_6) + M(p_8) + M(p_9) + M(p_{10}) + M(p_{12}) + M(p_{13}) + M(p_{15}) + M(p_{17}) + M(p_{21}) \leq 1$;
3) $M(p_{29}) + M(p_{30}) + M(p_{32}) + M(p_{33}) + M(p_{34}) + M(p_{35}) \geq 5$;
4) $M(p_{30}) + M(p_{32}) + M(p_{33}) + M(p_{34}) + M(p_{35}) \geq 4$.

For (N_1, M_{10}), the number of reachable markings under these four constraints is 39, which means that the number of reachable markings under W_{08} is no more than 39. The numbers of reachable markings under H_{13} and MBA are 167 and 1408,

respectively. Obviously, MBA admits much more markings than W_{08} and H_{13}. The computational complexity of H_{13} is NP-hard, while MBA is of polynomial complexity. Besides, we find that all reachable markings under H_{13} and W_{08} are reachable under MBA. One of the reachable markings of (N_1, M_{10}) under MBA is $M = 8p_1 + 8p_5 + 6p_{19} + p_{20} + p_{21} + 2p_{25} + p_{26} + p_{27} + 2p_{28} + p_{29} + p_{30} + p_{31} + p_{32} + p_{33}$, which is not admitted by W_{08} since it violates its third constraint. Also, it is not admitted by H_{13} since it violates $M(p_{14}) + M(p_{20}) + M(p_{21}) + M(p_{22}) + M(p_{23}) \leq 1$, which is one of the five constraints of H_{13}.

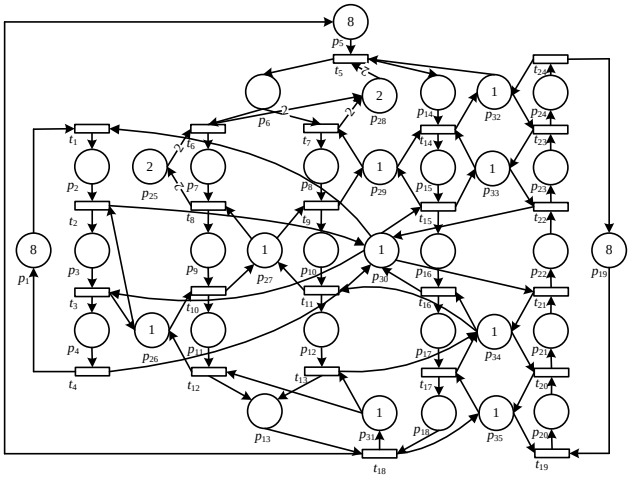

Fig. 3. A manufacturing system in [12].

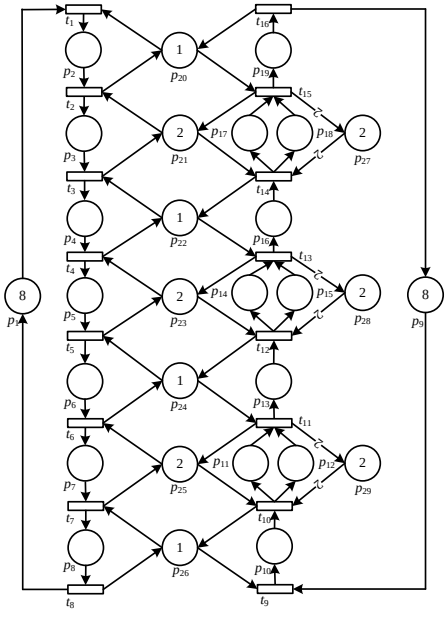

Fig. 4. A manufacturing system in [13].

The second illustrative manufacturing system comes from [13]. Its PN model is denoted as (N_2, M_{20}), which is shown in Fig. 4. For (N_2, M_{20}), the first two conditions in W_{08} can be changed to three constraints as follows:

1) $M(p_{10}) + M(p_{11}) + M(p_{13}) + M(p_{14}) + M(p_{16}) + M(p_{17}) + M(p_{19}) \leq 1$;

2) $M(p_2) + M(p_3) + M(p_4) + M(p_5) + M(p_6) + M(p_7) + M(p_8) + M(p_{10}) + M(p_{11}) + M(p_{13}) + M(p_{14}) + M(p_{16}) + M(p_{17}) + M(p_{19}) \leq 2$;

3) $M(p_2) + M(p_4) + M(p_6) + M(p_8) + M(p_{10}) + M(p_{13}) + M(p_{16}) + M(p_{19}) \leq 1$.

For (N_2, M_{20}), the number of reachable markings under these three constraints is 66, meaning that the number of reachable markings under W_{08} is no more than 66. The numbers of reachable markings of (N_2, M_{20}) under H_{15} and MBA are 444 and 5098, respectively. Hence, MBA admits much more markings than W_{08} and H_{15}. All markings reachable under W_{08} are reachable under MBA. 440 out of 444 reachable markings under H_{15} are reachable under MBA. One of the reachable markings of (N_2, M_{20}) under H_{15} is $M_1 = 7p_1 + p_6 + 6p_9 + p_{10} + p_{11} + p_{12} + p_{20} + 2p_{21} + p_{22} + 2p_{23} + p_{25} + 2p_{27} + 2p_{28}$, which is not admitted by MBA since no activity is terminable or mountable at M_1. This inspires an interesting issue of how to make MBA admit such kind of markings, to be investigated in our future work.

To further test the effectiveness of MBA, we have applied it to a large example and a railway network system studied in [9]. Moreover, the runtime of MBA has been tested. Details about them are available at https://github.com/luojianchao/DAP4FMSwithAssemblyOperations.git.

B. Comparative Studies

TABLE I
COMPARISON RESULTS OF EXISTING POLICIES.

Policies	Characteristics			
	Can it be used to AMSs with multiple resource acquisition?	Can it be used to AMSs with assembly operations?	Deadlock avoidance (A) or prevention (P)?	Can it be used to EAPN?
B_{1990} [2]	no	no	A	no
C_{2011} [3]	no	no	P	no
E_{2002} [6]	yes	no	A	no
E_{2006} [7]	yes	no	A	no
F_{2002} [8]	no	yes	P	no
F_{2003} [9]	no	no	P	no
F_{2017} [10]	no	no	P	no
H_{2013} [12]	yes	yes	P	no
H_{2015} [13]	yes	yes	P	no
H_{2001} [14]	no	no	P	no
J_{1995} [15]	yes	no	P	no
J_{1998} [17]	no	no	A	no
P_{2001} [22]	yes	no	A	no
P_{2008} [23]	no	no	P	no
R_{2004} [24]	yes	yes	A	no
T_{2005} [27]	yes	no	P	no
W_{2008} [29]	yes	yes	A	no
X_{1996} [31]	no	no	A	no
MBA	yes	yes	A	yes

This subsection makes brief comparative studies of deadlock control policies of AMSs in the existing literature. The results are shown in Table I. From it, we know that only MBA can be applied to EAPN. By Statement 1, we know that EAPN contains almost all existing manufacturing system-oriented Petri net models. Thus, MBA can be applied to many if not all manufacturing systems in the existing literature. However, other policies can only be applied to a subclass of EAPN. It means that MBA owns the highest universality among existing control policies.

From Table I, we know that only the policies in [12], [13], [24], [29] can be applied to AMSs with assembly operations and multiple resource acquisition. From the experimental

results in the last section, we know that MBA owns the highest permissiveness among them. Besides, MBA owns higher permissiveness than the policy in [9], which can be used to railway network systems.

Since the deadlock control method in [3] needs to generate the reachability graph of a system, it can only be applied to small scale systems. The methods in [14], [23] need to compute the set of strict minimum siphons, thus they can hardly handle large scale systems. The deadlock control methods in [12], [13], [22] need to solve a few integer linear programming problems. Thus, they are theoretically exponential with regard to the system scale. MBA is proved to be of polynomial complexity. Thus, it can handle large scale systems. More comparisons are needed to explore other advantages of MBA with respect to existing policies, which is in our future research agenda.

V. Conclusions

This work studies the deadlock avoidance problem of automated manufacturing systems allowing flexible routes, assembly operations, and multiple resource acquisition. To characterize all these features, we define a new class of Petri nets called EAPN. It can properly contain many if not all existing manufacturing system-oriented Petri net models. Based on it, a polynomial deadlock avoidance policy is proposed. As no existing deadlock control policy has been proposed to cope with EAPNs, it is tested through two existing automated manufacturing systems where a product may be made of multiple types of parts with the number of parts of the same type is limited by one. The obtained results show that the proposed new policy is much more permissive than the existing ones. Then, a large AMS is constructed to test the runtime of the proposed deadlock avoid policy. Experimental result shows that it is able to deal with large scale AMSs with the number of transitions up to 600 and the average number of activities in P up to 140. More research is needed to find the optimal deadlock avoidance or prevention policy for EAPN.

Acknowledgment

The authors would like to thank the editor, associate editor, and all anonymous reviewers for their thoughtful comments and suggestions that greatly helped improve the presentation and technical quality of this paper. This work is supported by the Fundamental Research Funds for the Central Universities under Grant 3102017OQD110, the Aviation Science Foundation of China under Grant 2015ZD53050, and the National Defense Basic Scientific Research Program under Grant N2016KD0003.

References

[1] K. Barkaoui, A. Chaoui, and B. Zouari, "Supervisory control of discrete event systems based on structure theory of Petri nets," *in Proc. IEEE Int. Syst., Man, Cybern. Comp. Cybern. Simul.*, Orlando, USA, 1997, pp. 3750-3755.

[2] Z. A. Banaszak, and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 724-734, Dec. 1990.

[3] 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, May. 2011.

[4] F. Chu and X. L. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 793-804, Dec. 1997.

[5] E. W. Dijkstra, *Co-Operating Sequential Processes*. New York: Academic, 1965, Programming Languages.

[6] J. Ezpeleta, F. Tricas, F. Garcia-Valles, and J. M. Colom, "A Banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states", *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 621-625, Aug. 2002.

[7] J. Ezpeleta and R. Valk, "A polynomial deadlock avoidance method for a class of nonsequential resource allocation systems," *IEEE Trans. Syst., Man, Cybern. A. Syst., Humans*, vol. 36, no. 6, pp. 1234-1243, Nov. 2006.

[8] M. P. Fanti, G. Maione, and B. Turchiano, "Design of supervisors to avoid deadlock in flexible assembly systems," *Int. J. Flex. Manuf. Syst.*, vol. 14, pp. 157-175, 2002.

[9] M. P. Fanti, A. Giua, and C. Seatzu, "A deadlock prevention method for railway networks using monitors for colored Petri nets," in *Proc. IEEE Int. Conf. on Syst., Man., Cybern.*, Washington, USA, Oct. 2003, pp. 1866-1873.

[10] Y. X. Feng, K. Y. Xing, Z. X. Gao, and Y. C. Wu, "Transition cover-based robust Petri net controllers for automated manufacturing systems with a type of unreliable resources," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 3019-3029, Nov. 2017.

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

[12] H. 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.

[13] H. 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.

[14] Y. S. Huang, M. D. Jeng, X. L. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 3, pp. 283-305, Jan. 2001.

[15] M. D. Jeng, and F. Dicesare, "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 3, pp. 317-327, 1995.

[16] S.-D. Lang, "An extended Banker's algorithm for deadlock avoidance," *IEEE Trans. Software Eng.*, vol. 25, pp. 428-432, May/June 1999.

[17] M. Lawley, S. Reveliotis, and P. Ferreira, "The application and evaluation of Banker's algorithm for deadlock-free buffer allocation in flexible manufacturing systems," *Int. J. Flexible Manuf. Syst.*, vol. 10, pp. 73-100, 1998.

[18] M. Lawley, "Integrating flexible routing and algebraic deadlock avoidance policies in automated manufacturing systems," *Int. J. Prod. Res.*, vol. 38, no. 13, pp. 2931-2950, Mar. 2000.

[19] G. Y. Liu, Z. Li, K. Barkaoui, and A. M. Al-Ahmari, "Robustness of deadlock control for a class of Petri nets with unreliable resources," Information Sciences, vol. 235, no. 6, pp. 259-279, Jun. 2013.

[20] G. Mejía, J. P. Caballero-Villalobos, and C. Montoya, "Petri nets and deadlock-free scheduling of open shop manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 6, pp. 1017-1028, Jun. 2018.

[21] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541-580, Apr. 1989.

[22] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. Autom. Control*, vol. 46, no. 10, pp. 1572-1583, Oct. 2001.

[23] 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, pp. 1337-1348, Nov. 2008.

[24] E. Roszkowska, "Supervisory control for deadlock avoidance in compound processes," *IEEE Trans. Syst., Man, Cybern., A, Syst., Humans*, vol. 34, no. 1, pp. 52-64, Jan. 2004.

[25] N. Ran, H. Su, and S. Wang, "An Improved Approach to Test Diagnosability of Bounded Petri Nets," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 297-303, Jan. 2017.

[26] E. Roszkowska, "Deadlock avoidance in concurrent compound pipeline processes," *Arch. Inform. Teoret. Stosowanej*, vol. 2, pp. 227-242, 1990.

[27] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," *in Proc. Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 272-278.

[28] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models," *IEEE Trans. Robot. Autom. Mag.*, vol. 6, no. 6, pp. 713-723, Dec. 1990.

[29] N. Q. Wu, M. Zhou, and Z. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Trans. Syst., Man, Cybern., A, Syst., Humans*, vol. 38, no. 1, pp. 56-69, Jan. 2008.

[30] D. Xiang, G. Liu, C. Yan, C. Jiang, "Detecting data inconsistency based on the unfolding technique of Petri nets," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 2995-3005, Dec. 2017.

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

[32] H. S. Hu, and Y. Liu, "Supervisor Synthesis and Performance Improvement for Automated Manufacturing Systems by Using Petri Nets," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 450-458, Apr. 2015.

[33] F. J. Yang, N. Q. Wu, K. Z. Gao, C. J. Zhang, Y. T. Zhu, R. Su, and Y. Qiao, "Efficient approach to cyclic scheduling of single-arm cluster tools with chamber cleaning operations and wafer residency time constraint," *IEEE Trans. Semicond. Manuf.*, doi: 10.1109/TSM.2018.2811125, 2018.