Deadlock-Free Scheduling of Flexible Assembly Systems Based on Petri Nets and Local Search

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

***Abstract*—Deadlock-free scheduling and control is critical for optimizing the performance of flexible assembly systems (FASs). Based on the Petri net models of FASs, this paper integrates a deadlock prevention policy with local search and develops a novel deadlock-free scheduling algorithm. A solution of the scheduling problem is coded as a chromosome representation that is a permutation with repetition of parts. By using the deadlock prevention policy, a repairing algorithm (RA) is developed to repair unfeasible chromosomes. A perturbation strategy based on *estimation of distribution* algorithm is developed to escape from local optima. Moreover, to improve population diversity, an acceptance criterion (AC) based on Pareto dominance is proposed. The chromosome representation, RA, perturbation strategy, and AC together support the cooperative aspect of local search for scheduling problems strongly.**

***Index Terms*—Deadlock prevention policy, flexible assembly system (FAS), local search, Petri net (PN), scheduling.**

Nomenclature

N	$(P \cup P_s \cup P_e \cup P_R, T, F)$, an assembly PN.
P	Set of activity (operation) places.
P_s	Set of start activity places.
P_e	Set of end activity places.
P_R	Set of resource places.
$[N]$	Incidence matrix of N.
$\mathbb{R}(N, M)$	Set of all reachable markings of N from M.
$M(S)$	$\Sigma_{p \in S} M(p)$, where S is a set of places.
$\mathbb{Z}$	Set of integers.
$\mathbb{N}$	$\{0, 1, 2, \ldots\}$.
$\mathbb{N}^+$	$\{1, 2, 3, \ldots\}$.
$\mathbb{N}_k$	$\{1, 2, \ldots, k\}$.
$\Vert\Delta\Vert$	Support of a P-vector Δ in N.
N^{-1}	$(P \cup P_s \cup P_e \cup P_R, T, F^{-1})$, reverse of N.
F^{-1}	$\{(x, y) \mid (y, x) \in F\}$, the set of arcs in N^{-1}.
$f(x)$	A vertex $x \in P \cup P_s \cup P_e \cup P_R \cup T$ in N^{-1}.
$d(p)$	Processing time of the activity modeled by p.
$\wp(\pi)$	Set of places in A-chain π.
$\Im(\pi)$	Set of transitions in A-chain π.
$R(\pi)$	Set of all resource types required by places in $\wp(\pi)$.
span(s)	Makespan of schedule s.
$o(\Gamma)$	Sequence of activities corresponding to chromosome Γ.
$\alpha(\Gamma)$	Sequence of transitions corresponds to Γ.

Manuscript received February 25, 2018; revised April 13, 2018; accepted June 28, 2018. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 3102017OQD110, in part by the Aviation Science Foundation of China under Grant 2015ZD53050, in part by the National Defense Basic Scientific Research Program under Grant N2016KD0003, and in part by the National Natural Science Foundation of China under Grant 61573278. This paper was recommended by Associate Editor M. K. Tiwari. *(Corresponding authors: ZhiQiang Liu; MengChu Zhou.)*

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

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA and also with the Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China (e-mail: zhou@njit.edu).

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

Digital Object Identifier 10.1109/TSMC.2018.2855685

I. Introduction

An AUTOMATED manufacturing system (AMS) is a computer-controlled system that exhibits a high degree of resource sharing. It consists of a finite set of resources and can process different kinds of parts based on a prescribed sequence of operations. Its interacting parts and shared resources may lead to deadlock states under which the system remains indefinitely blocked and cannot terminate its tasks. Therefore, developing efficient control and scheduling algorithms to avoid deadlocks while optimizing system performance is significant.

The deadlock control problem of AMSs without assembly processes has been extensively studied, and many deadlock control policies have been proposed [3], [4], [7], [9], [10], [17], [19], [20], [22], [29], [31], [32], [36], [38], [40], [42], [45]–[47], [60]. More detailed review can be found in [48]. These policies can ensure the deadlock-free operation of AMSs. As they do not take the processing time into consideration, their operational performance is not guaranteed. Thus, deadlock-free scheduling of such systems is still an open field. Because of its NP-hard nature [2], [41], only a few researchers address it [1], [2], [11], [21], [24], [33]–[35], [37], [41], [61]. Ramasway and Joshi [24] provide a mathematical model for deadlock-free scheduling problems of AMSs with material handing devices and limited buffers and use a Lagrangian relaxation heuristic to simplify the model to search for the optimized average flow time. Abdallah *et al.* [1] use timed Petri nets (PNs) to model AMSs and propose a deadlock-free scheduling algorithm. It generates a partial reachability graph to find the optimal or near-optimal deadlock-free schedule. Gang and Wu [37] develop a genetic algorithm based on PN with infinity buffers to find a schedule, analyze deadlocks that may occur with the schedule, and add some necessary buffers to avoid the deadlocks. Wu and Zhou [33] develop colored-timed resource-oriented PNs to model AMSs that are failure-prone. Based on the developed model and a deadlock control policy, heuristic rules are proposed to schedule the system in real time. Xing *et al.* [41] embed a deadlock avoidance policy into a genetic algorithm and develop a deadlock-free genetic algorithm for AMSs. Based on the deadlock search algorithm in [40], a one-step look-ahead method is developed to amend infeasible solutions into feasible ones. The latest scheduling algorithms for AMSs can be found in [62]–[66].

In contrast, few researchers have addressed the deadlock control problems in flexible assembly systems (FASs) [8], [12]–[16], [26], [30], [39]. Assembly is an important process of many manufacturing systems, which puts together two or more parts to produce an intermediate or final product. Fanti *et al.* [8] address deadlock avoidance problems in FASs from a supervisory control point of view. Two supervisors characterized by easy implementation, efficiency, and flexibility in resource management are proposed. Roszkowska [26] contribute a method for deadlock avoidance in AMSs with assembly/disassembly processes. A supervisor is proposed for a subclass of realizable systems to avoid deadlocks. Hsieh [12]–[14] studies FASs with unreliable resources. He first analyzes the fault tolerant properties of the systems and presents the conditions under which the operation of the system can be maintained in the presence of resource failures [12], [14]. Then, he studies the deadlock avoidance problem for FASs with alternative routes, proposes a deadlock avoidance algorithm with polynomial complexity, and accesses its robustness with respect to resource failures. Wu *et al.* [30] studies the deadlock avoidance problem for FASs with assembly/disassembly material flow and resource-oriented PNs. A deadlock control policy is proposed and proved to be computationally efficient and less conservative than existing ones [26]. Xing *et al.* [39] focus on the deadlock prevention problem for FASs. Two kinds of structural objects that can lead to an empty siphon and cause deadlocks are proposed. By preventing such structural objects from causing FAS deadlocks, a PN controller is proposed. An illustrative example shows that it is more permissive than those in [26] and [30].

Although several control policies for FASs have been proposed, to the authors' best knowledge, no existing scheduling algorithm dealing with the deadlock-free scheduling problem of FASs has been proposed. This may be because:

1) the scheduling problem of AMSs with assembly processes is more difficult than that without them since parts waiting for the assembly with other parts may also lead to deadlocks, and hence it is also NP-hard;
2) combining control policies with scheduling algorithms poses a new challenge since deadlock control of FASs is not addressed widely.

Metaheuristic algorithms, e.g., genetic algorithm [58] and local search algorithm [49], have received extensive research. Genetic algorithms have been well applied in scheduling of flexible manufacturing systems [11], [41]. Detailed review can be found in [59]. They use a crossover operation to obtain better chromosomes (solutions) by exchanging information in parent chromosomes and a mutation operation to enhance the biodiversity of the population. Local search algorithms use a local search operation to search the neighbors of incumbent chromosomes to find better ones and a perturbation operation perturbs locally optimal solutions to escape from local optima. Thus, a genetic algorithm and local search algorithm are different. The latter has solved many complex and large problems successful [5], [23], [51]–[53]. However, no work has been done for the deadlock-free scheduling problem of FASs using it.

This paper intends to fill this gap by proposing a local search algorithm for the scheduling problem of deadlock-prone FASs. We assume that each part in our system has only one processing route. The reason why we make such an assumption is twofold. First, the deadlock-free scheduling problem of FAS with or without flexible routing has not been studied before. Beginning a problem from FAS without flexible routing still represents an important step in this area. Second, existing deadlock control policies for FAS with flexible routing constrain the system too much. Integrating them with scheduling algorithms tends to affect the scheduling performance greatly. Thus, research on scheduling FAS with flexible routing is delayed to the future. We use place-timed PNs to model the systems. The optimization objective is to minimize the total completion time or makespan. To embed an existing deadlock prevention policy into a scheduling algorithm, the policy is redefined in a concise way that is analogous to the original one, but can be easily embedded into a scheduling algorithm. Based on the built PN model and redefined deadlock prevention policy, a deadlock-free local search scheduling algorithm is developed. In order to enhance the global search ability and keep the population diversity of local search, a perturbation policy based on the estimation of distribution algorithm (EDA) and an acceptance criterion (AC) based on Pareto dominance for selection are proposed. Our proposed deadlock-free local search scheduling algorithm is suitable for deadlock-prone FASs with shared resources, lot size, and concurrency.

The rest of this paper is organized as follows. Section II introduces the PN modeling of FASs. A PN-based deadlock prevention policy is reviewed and redefined in Section III. Section IV establishes a deadlock-free scheduling algorithm by integrating the redefined policy with local search. The effectiveness of the proposed algorithm is illustrated in Section V. Section VI concludes this paper.

II. Basic PN Definitions and FAS Scheduling Models

First, we review the basics of PNs. More details can be found in [28], [43], [44], and [55]–[57]. Next, we describe the PN model of FAS for scheduling.

A. Basic Definitions of PNs

A PN is a 4-tuple $N = (P, T, F, W)$, where P and T are finite and disjoint sets. P is a set of places, and 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. $W : F \to \mathbb{N}^+$ is a mapping that assigns to each arc a positive integer or weight. N is ordinary if $\forall(x, y) \in F$, $W(x, y) = 1$, and in this case, W can be omitted.

Given a vertex $x \in P \cup T$, its preset and postset 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 $\alpha = x_1x_2\ldots x_k$, where $x_i \in P \cup T$ and $(x_i, x_{i+1}) \in F$ $\forall i \in \mathbb{N}_{k-1}$.

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

A transition $t \in T$ is enabled at a marking M, denoted as $M[t>$, if $\forall p \in {}^\bullet t$, $M(p) \geq W(p, t)$. An enabled transition t at M can fire, generating a new marking M', denoted as $M[t> M'$, where $M'(p) = M(p) - W(p, t)$ $\forall p \in {}^\bullet t \backslash t^\bullet$, $M'(p) = M(p) + W(t, p)$ $\forall p \in t^\bullet \backslash {}^\bullet t$, and otherwise, $M'(p) = M(p) - W(p, t) + W(t, p)$. A sequence of transitions $\alpha = t_1t_2\ldots t_k$ is feasible from a marking M if there exists $M_i[t_i> M_{i+1}$, $i \in \mathbb{N}_k$, where $M_1 = M$. We call that M_i is reachable from M. Let $\mathbb{R}(N, M)$ denote the set of all reachable markings of N from M.

The incidence matrix of N is a mapping $[N] : P \times T \to \mathbb{Z}$ such that $[N](p, t) = -W(p, t)$ $\forall p \in {}^\bullet t \backslash t^\bullet$; $[N](p, t) = W(t, p)$ $\forall p \in t^\bullet \backslash {}^\bullet t$; otherwise $[N](p, t) = W(t, p) - W(p, t)$. A nonzero $|P|$-vector Δ: $P \to N$ is a P-invariant if $\Delta^T \cdot [N] = 0$. The support of a P-invariant is the set of places $\|\Delta\| = \{p \in P \mid \Delta(p) \neq 0\}$. Let Δ be a P-invariant and $M \in \mathbb{R}(N, M_0)$, then $\Delta^T \cdot M = \Delta^T \cdot M_0$.

A nonempty set of places $S \subseteq P$ is a siphon if ${}^\bullet S \subseteq S^\bullet$. Let S be a siphon and $M \in \mathbb{R}(N, M_0)$. If $M(S) = 0$, then $\mathbb{M}'(S) = 0$ $\forall M' \in \mathbb{R}(N, M)$. A transition t is *live* if $\forall M \in \mathbb{R}(N, M_0)$, there exists a marking $M' \in \mathbb{R}(N, M)$ such that t is enabled at M'. A transition t is *dead* at $M \in \mathbb{R}(N, M_0)$ if there exists no marking $M' \in \mathbb{R}(N, M)$ such that t is enabled at M'. A marked PN is live if all transitions are live.

Let $(N_1, M_{10}) = (P_1, T_1, F_1, W_1, M_{10})$ and $(N_2, M_{20}) = (P_2, T_2, F_2, W_2, M_{20})$ be two marked PNs. (N_1, M_{10}) and (N_2, M_{20}) are compatible if $\forall p \in P_1 \cap P_2$, $M_{10}(p) = M_{20}(p)$ and $\forall(x, y) \in F_1 \cap F_2$, $W_1(x, y) = W_2(x, y)$. In this case, they can be composed and their composition is a marked net $(N_1, M_{10}) \otimes (N_2, M_{20}) = (N_1 \otimes N_2, M_0)$, where $N_1 \otimes N_2 = (P_1 \cup P_2, T_1 \cup T_2, F_1 \cup F_2, W)$ where $\forall(x, y) \in F_i$, $W(x, y) = W_i(x, y)$, $i \in \{1, 2\}$ and $\forall p \in P_i$, $M_0(p) = M_{i0}(p)$, $i \in \{1, 2\}$.

The reverse of $N = (P, T, F, W)$ is a PN, denoted as $N^{-1} = (P, T, F^{-1}, W^{-1})$, where $F^{-1} = \{(x, y) \mid (y, x) \in F\}$ and $W^{-1}(x, y) = W(y, x)$. To distinguish a vertex $x \in P \cup T$ in N and N^{-1}, it is denoted as $f(x)$ in N^{-1}.

B. Place-Timed PN Scheduling Models of FASs

The FAS studied in this paper consists of m type of resources and can manufacture and assemble l types of products. Let $R = \{r_i \mid i \in \mathbb{N}_m\}$ be the set of resource types. Each type of resources may be a kind of workstations, buffers, or robots. The capacity of r_i is a positive integer, denoted as C_i. A product is obtained from some raw parts through a series of manufacturing and assembly operations or activities. In a manufacturing activity, parts are only processed, while in an assembly, two or more parts are assembled into a new (intermediate or final) part. The same raw parts can only be used to assemble a type of products. Let n be the number of part types, and $J = \{J_i \mid i \in \mathbb{N}_n\}$ be the sets of part types, and Ψ_i be the number of type-i parts to be processed.

A processing route of a part is a sequence of manufacturing and assembly activities. A part of J_i has only one processing route, denoted as w_i. Route w_i can be expressed as $w_i = o_{is}o_{i1}o_{i2}\ldots o_{iL_i}o_{ie}$, where o_{ij} is the jth manufacturing or assembly activity in w_i, L_i is the total number of manufacturing and assembly activities in w_i, and o_{is} (o_{ie}) denote the start (end) activity of J_i.

In our PN model, the processing route of part J_i, i.e., w_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} and p_{ie} represent the activities o_{is} and o_{ie}, respectively, p_{ij} is an activity place representing activity o_{ij}, t_{ij} represents the start of o_{ij}, and $t_{i(j+1)}$ represents the completion of o_{ij}. Hence, the marked PN model of w_i can be denoted as

$$N_i = (P_i \cup \{p_{is}, p_{ie}\}, T_i, F_i, M_{i0}), i \in \mathbb{N}_n$$

where $P_i = \{p_{i1}, \ldots, p_{iL_i}\}$, $T_i = \{t_{i1}, t_{i2}, \ldots, t_{i(L_i+1)}\}$, and $F_i = \{(p_{is}, t_{i1}), (t_{i1}, p_{i1}), \ldots, (p_{iL_i}, t_{i(L_i+1)}), (t_{i(L_i+1)}, p_{ie})\}$. M_{i0} is the initial marking, $M_{i0}(p) = 0$ $\forall p \in P_i \cup \{p_{ie}\}$, and $M_{i0}(p_{is}) = \Psi_i$.

The processing routes of different parts may share some identical activities. For simplicity, the identical activities are merged as one, i.e., they have the same activity place, start and completion transition.

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, suppose that each activity requires only one resource. Let $R(p)$ denote the resource required by activity place p. Then, add arcs from $R(p)$ to each transition in ${}^\bullet p$, denoting the allocation of $R(p)$, and arcs from each transition in $p^\bullet$ to $R(p)$, denoting the release of $R(p)$. Let F_R denote the set of arcs related with resource places. The whole system can be modeled by the following marked PN:

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

where $P = \{P_i \mid i \in \mathbb{N}_n\}$, $P_s = \{p_{is} \mid i \in \mathbb{N}_n\}$, $P_e = \{p_{ie} \mid i \in \mathbb{N}_n\}$, $T = \{T_i \mid i \in \mathbb{N}_n\}$, $F = F_Q \cup F_R$, and $F_Q = \{F_i \mid i \in \mathbb{N}_n\}$. P_s, P_e, and P_R are finite and disjoint sets, i.e., $P_s \neq \varnothing$, $P_e \neq \varnothing$, $P_R \neq \varnothing$, and $P_s \cap P_e \cap P_R = \varnothing$. 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 $M_0(r_i) = C_i$ $\forall r_i \in P_R$.

When all activities of all parts are finished, N reaches a final marking, denoted as M_f, which is defined as follows. $M_f(p_{ie}) = M_0(p_{is})$ $\forall p_{ie} \in P_e$, $M_f(p) = 0$ $\forall p \in P \cup P_s$, and $M_f(r_i) = C_i$ $\forall r_i \in P_R$. For a marking $M \in \mathbb{R}(N, M_0)$, M is *safe*

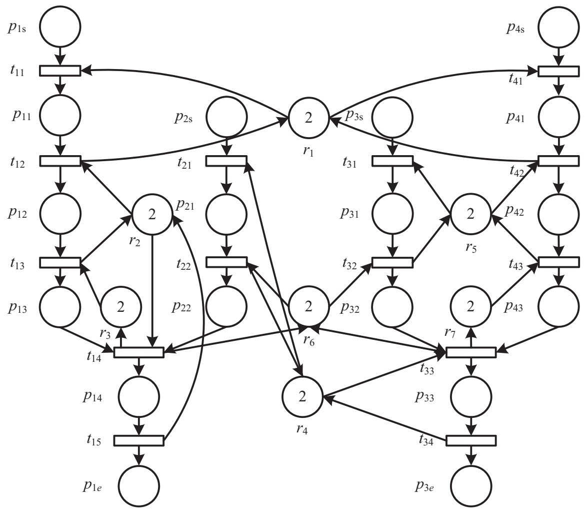


Fig. 1. APNS of the FAS in Example 1.

if $M_f \in \mathbb{R}(N, M)$; and otherwise, M is *unsafe*. In this paper, we suppose that M_0 is safe.

For $p \in P$, let $d(p)$ denote the processing or assembly time of the activity modeled by p, and particularly, for $p \in P_s \cup P_e \cup P_R$, $d(p) = 0$. In the following, we refer to the above place-timed PN model as an assembly PN for scheduling (APNS). Let us illustrate the modeling method with the following example.

Example 1: Consider an FAS with seven types of resources r_1–r_7, i.e., $R = \{r_i \mid i \in \mathbb{N}_7\}$. $C_i = 2\ \forall i \in Z_7$. The system can manufacture four types of parts and assemble two types of products. The part type set is $J = \{J_i \mid i \in \mathbb{N}_4\}$. Type-1 parts are manufactured on r_1, r_2, and r_3 and type-2 parts on r_4 and r_6 sequentially. Then type-1 and type-2 parts are assembled on r_2. Type-3 parts are manufactured on r_5 and r_6 and type-4 parts on r_1, r_5, and r_7 sequentially. Then type-3 and type-4 parts are assembled on r_4. The processing routes of type-1$\sim$4 parts are $w_1 = o_{1s}o_{11}o_{12}o_{13}o_{14}o_{1e}$, $w_2 = o_{2s}o_{21}o_{22}o_{23}o_{2e}$, $w_3 = o_{3s}o_{31}o_{32}o_{33}o_{3e}$, and $w_4 = o_{4s}o_{41}o_{42}o_{43}o_{44}o_{4e}$, respectively. The processing routes of type-1$\sim$4 are modeled by $p_{1s}t_{11}p_{11}t_{12}p_{12}t_{13}p_{13}t_{14}p_{14}t_{15}p_{1e}$, $p_{2s}t_{21}p_{21}t_{22}p_{22}t_{14}p_{14}t_{15}p_{1e}$, $p_{3s}t_{31}p_{31}t_{32}p_{32}t_{33}p_{33}t_{34}p_{3e}$, and $p_{4s}t_{41}p_{41}t_{42}p_{42}t_{43}p_{43}t_{33}p_{33}t_{34}p_{3e}$, respectively. Since o_{14} and o_{23} represent the same activity, their activity places and start and completion transitions are the same. So are o_{1e} and o_{2e}, o_{33} and o_{44}, o_{3e} and o_{4e}. Then, $P = \{p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{31}, p_{32}, p_{33}, p_{41}, p_{42}, p_{43}\}$, $P_s = \{p_{1s}, p_{2s}, p_{3s}, p_{4s}\}$, $P_e = \{p_{1e}, p_{3e}\}$, and $P_R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$. The APNS model of the whole system is shown in Fig. 1, where the numbers of raw parts to be processed are not given.

For a transition $t \in T$, ${}^{(r)}t$ and $t^{(r)}$ denote its input and output resource place sets, respectively, and ${}^{(a)}t$ and $t^{(a)}$ denote its input and output activity place sets, respectively. For a set with a single element, e.g., ${}^{(a)}t = \{p\}$, we use ${}^{(a)}t = p$ for simplicity. In APNS $\forall p \in P \cup P_s \cup P_e$, $|{}^{\bullet}p| = |p^{\bullet}| = 1$ and $\forall t \in T$, $|t^{(a)}| \leq 1$. $t \in T$ is an assembly transition if $|{}^{(a)}t| > 1$. Let T_s denote the set of all transitions whose input activity places are in P_s, i.e., $T_s = \{t \in T \mid {}^{(a)}t \in P_s\}$. For the APNS in Fig. 1, $T_s = \{t_{11}, t_{21}, t_{31}, t_{41}\}$.

For a given marking $M \in \mathbb{R}(N, M_0)$, $t \in T$ is activity-enabled at M if $M(p) > 0\ \forall p \in {}^{(a)}t$, t is resource-enabled at M if ${}^{(r)}t = \varnothing$ or $M({}^{(r)}t) > 0$. In an APNS, only transitions, which are activity- and resource-enabled at the same time, can fire. For a resource $r \in P_R$, let $H(r)$ denote the set of places whose activities require resource r, i.e., $H(r) = \{p \in P \mid R(p) = r\}$. These notations can be extended to a set. For example, $R' \subseteq P_R$, $H(R') = \{p \in P \mid R(p) \in R'\}$. For any marking $M \in \mathbb{R}(N, M_0)$, $M(H(R') \cup R') = M_0(R')$. That means, $H(R') \cup R'$ is the support of a P-invariant of N.

Given an APNS (N, M_0), a sequence of transitions s is named as a schedule if $M_0[s> M_f$. Let $S(N, M_0) = \{s \mid M_0[s> M_f\}$ denote the set of all schedules of (N, M_0). Let $s = t_1t_2\ldots t_k$ be a schedule in $S(N, M_0)$ and $f(t_l[o_{ij}])$ denote the firing time of t_l that is the start time of activity o_{ij}. In APNS, $t_l[o_{ij}]$ can fire only after activity $o_{i(j-1)}$ is finished (according to the prescribed activity sequence) and $t_{(l-1)}$ fires (according to the order in s). Suppose that $t_{(l-1)}$ corresponds to o_{uv} and t_q corresponds to $o_{i(j-1)}$. Then $f(t_l[o_{ij}]) = \max\{f(t_q[o_{i(j-1)}]) + d(t_q^{(a)}), f(t_{(l-1)}[o_{uv}])\}$ and the makespan of s is $span(s) = \max_{i,j,l}\{f(t_l[o_{ij}]) + d(t_l{}^{(a)})\}$. The scheduling problem of (N, M_0) is to find a schedule in $S(N, M_0)$ such that its makespan is as small as possible.

III. Deadlock Prevention Policies of Marked APNSs

According to the structures of APNS and the assembly PN (APN) models [39] for FASs, we know that deadlock prevention policies for APN are available for APNS of the same FAS. Hence, the deadlock prevention policy for APN proposed in [39] can be used for APNS.

The deadlock prevention policy in [39] has not considered the integration with any scheduling algorithms. To integrate it with a scheduling algorithm, we have to redefine it. The deadlock prevention policy in [39] contains the controllers for all A-circuits and closed Ω-structures. To redefine it, we have to introduce the definitions and theorems related with A-circuits and closed Ω-structures first. Then, we redefine the controllers for A-circuits and closed Ω-structures and give the proofs on their correctness.

A. Structural Characteristics and Deadlock Analysis of APNSs

Xing *et al.* [39] present two kinds of structural objects that may cause deadlocks in APNS, i.e., *A-circuit* and Ω-*structure*. Their definitions are as follows. Let $(N, M_0) = (P \cup P_s \cup P_e \cup P_R, T, F, M_0)$ be an APNS.

Definition 1 [39]: An activity path (A-path) is a path $\pi = p_1t_1\ldots p_kt_k$ where $p_i \in P$ and $t_i \in T$, $i \in \mathbb{N}_k$.

Definition 2 [39]: An *A-chain* is recursively defined as follows.

1) An A-path is an A-chain.
2) Let $\pi_1 = p_{11}t_{11}\ldots p_{1k}t_{1k}$ and $\pi_2 = p_{21}t_{21}\ldots p_{2j}t_{2j}$ be two A-chains. If ${}^{(r)}t_{1k} = R(p_{21})$, then π_1 and

π_2 are said to be compatible, and $\pi = \pi_1\pi_2 = p_{11}t_{11}\ldots p_{1k}t_{1k}p_{21}t_{21}\ldots p_{2j}t_{2j}$ is an A-chain.

An A-path is an activity subroute of a path and starts from an activity place and ends with a transition. An A-chain is a sequence of A-paths, $\pi = \pi_1\pi_2\ldots\pi_{k-1}\pi_k$ such that π_i and π_{i+1}, $i \in \mathbb{N}_{k-1}$ are compatible.

Let $\pi = p_1t_1\ldots p_kt_k$ be an A-chain in N, $\wp(\pi) = \{p_1, \ldots, p_k\}$ and $\Im(\pi) = \{t_1, \ldots, t_k\}$ denote the sets of places and transitions in π, respectively, $R(\pi) = R(\wp(\pi))$ denote the set of all resource types required by the activities in π.

Definition 3 [39]: Let $\theta = \pi_1\pi_2\ldots\pi_{k-1}\pi_k = p_1t_1\ldots p_qt_q$ be an A-chain and $M \in \mathbb{R}(N, M_0)$. θ is *closed* if π_1 and π_k are compatible, i.e., ${}^{(r)}t_q = R(p_1)$. A closed A-chain is called as an *A-circuit*. An A-circuit θ is *saturated* under M if $M(\wp(\theta)) = M_0(R(\theta))$.

If two A-circuits θ_1 and θ_2 have the same set of resources, then they can be written as $\theta_1 = p_{11}t_{11}\ldots p_{1k}t_{1k}$ and $\theta_2 = p_{21}t_{21}\ldots p_{2j}t_{2j}$ with ${}^{(r)}t_{1k} = R(p_{11}) = R(p_{21}) = {}^{(r)}t_{2j}$, and $\theta = \theta_1\theta_2$ is also an A-circuit. Thus, for any A-circuit θ, there is a unique maximal A-circuit, denoted as $\Lambda(\theta)$, such that θ is a subcircuit of $\Lambda(\theta)$, $\wp(\theta) \subseteq \wp(\Lambda(\theta))$ and $R(\theta) = R(\Lambda(\theta))$. Let $O(N)$ denote the set of all maximal A-circuits in N.

Definition 4 [39]: Let $\pi_1 = p_{11}t_{11}\ldots p_{1k}t_{1k}$ be an A-chain and $\pi_2 = p_{21}t_{21}\ldots p_{2j}t_{2j}$ be an A-path. If $\wp(\pi_1) \cap \wp(\pi_2) = \varnothing$ and $t_{1k} = t_{2j}$ (an assembly transition), then π_1 and π_2 are said to be a *v-structure*, denoted as $v = (\pi_1, \pi_2)$. Furthermore, v-structure $v = (\pi_1, \pi_2)$ is *closed* if $R(p_{11}) = R(p_{21})$; and $R(p_{11})$ is called a *seal-resource* of v.

Note that a v-structure is not symmetrical, i.e., $v = (\pi_1, \pi_2)$ is a v-structure, while $v^T = (\pi_2, \pi_1)$ may be not a v-structure.

Definition 5 [39]: An *Ω-structure* is a sequence of alternating A-chains and A-paths $w = \pi_{11}\pi_{21}\ldots\pi_{1k}\pi_{2k}$ such that (π_{1i}, π_{2i}), $i \in \mathbb{N}_k$, is a v-structure and $R({}^*\pi_{2i}) = R({}^*\pi_{1(i+1)})$, $i \in \mathbb{N}_{k-1}$, where ${}^*\pi$ denotes the first place of π.

Let $w = \pi_{11}\pi_{21}\ldots\pi_{1k}\pi_{2k}$ be an Ω-structure, $\prod_1 = \{\pi_{1i}, i \in \mathbb{N}_k\}$, and $\prod_2 = \{\pi_{2i}, i \in \mathbb{N}_k\}$. Then w can be written as $(\prod_1, \prod_2)$ for simplicity. Let $\wp(\prod_i) = \{\wp(\pi) \mid \pi \in \prod_i\}$ and $R(\prod_i) = R(\wp(\prod_i))$, $i \in \{1, 2\}$. w is *closed* if $\wp(\prod_1) \cap \wp(\prod_2) = \varnothing$ and $R({}^*\pi_{11}) = R({}^*\pi_{2k})$; and $R({}^*\pi_{11})$ is called a *seal-resource* of w.

Note that a closed v-structure is a closed Ω-structure. Let $w_1 = (\prod_{11}, \prod_{12})$ and $w_2 = (\prod_{21}, \prod_{22})$ be two closed Ω-structures such that $R(\prod_{11}) = R(\prod_{21})$, $\prod_1 = \prod_{11} \cup \prod_{21}$, $\prod_2 = \prod_{12} \cup \prod_{22}$. If $\prod_1 \cap \prod_2 = \varnothing$, then w_1 and w_2 form a closed Ω-structure $w = (\prod_1, \prod_2)$. Thus, for a closed Ω-structure $w = (\prod_1, \prod_2)$, there is a unique maximal closed Ω-structure, or ϖ–structure for short, $\Lambda(w) = (\prod_{10}, \prod_{20})$, such that w is an Ω-substructure of $\Lambda(w)$, and $R(\prod_1) = R(\prod_{10})$. Let $W(N)$ denote the set of all ϖ–structures in N.

Definition 6 [39]: Let $M \in \mathbb{R}(N, M_0)$ and $w = (\prod_1, \prod_2)$ be a closed Ω-structure. w is *semi-saturated and semi-empty* at M if $M(\wp(\prod_1)) = M_0(R(\prod_1))$ and $M(\wp(\prod_2)) = 0$.

Example 2: Consider the APNS in Fig. 1. Let

$$\theta_1 = p_{12}t_{13}p_{13}t_{14},\ \theta_2 = p_{21}t_{22}p_{32}t_{33},\ \pi_1 = p_{31}t_{32}p_{32}t_{33}$$
$$\pi_2 = p_{42}t_{43}p_{43}t_{33},\ \pi_3 = p_{11}t_{12}p_{12}t_{13}p_{13}t_{14}$$
$$\pi_4 = p_{41}t_{42}p_{31}t_{32}p_{32}t_{33}p_{21}t_{22}p_{22}t_{14},\ \pi_5 = p_{22}t_{14}$$
$$\pi_6 = p_{32}t_{33},\ \pi_7 = p_{41}t_{42}p_{42}t_{43}p_{43}t_{33},\ \pi_8 = p_{31}t_{32}p_{22}t_{14}$$
$$\pi_9 = p_{41}t_{42}p_{31}t_{32}p_{32}t_{33},\ \pi_{10} = p_{41}t_{42}p_{31}t_{32}p_{22}t_{14},\ \text{and}$$
$$\pi_{11} = p_{41}t_{42}p_{42}t_{43}p_{43}t_{33}p_{21}t_{22}p_{22}t_{14}.$$

Then θ_1, π_1–π_3, π_5–π_7 are A-paths, while θ_2, π_4, and π_8–π_{11} are A-chains. θ_1 and θ_2 are both A-circuits and maximal too, and $O(N) = \{\theta_1, \theta_2\}$. $w_1 = (\pi_1, \pi_2)$, $w_2 = (\pi_2, \pi_1)$, $w_3 = (\pi_4, \pi_3)$, and $w_4 = (\pi_{11}, \pi_3)$ are v-structures. Since $R(p_{31}) = R(p_{42})$ and $R(p_{11}) = R(p_{41})$, w_1–w_4 are closed v-structures.

Let $w_5 = \pi_3\pi_5\pi_6\pi_7$, $w_6 = \pi_5\pi_3\pi_7\pi_6$, $w_7 = \pi_8\pi_3\pi_9\pi_2$, and $w_8 = \pi_7\pi_6\pi_5\pi_3\pi_{10}\pi_3$. Since (π_3, π_5) and (π_6, π_7) are v-structures, $R({}^*\pi_5) = R({}^*\pi_6) = r_6$ and $R({}^*\pi_3) = R({}^*\pi_7) = r_1$, $w_5 = \pi_3\pi_5\pi_6\pi_7$ is a closed Ω-structure. Similar, w_6–w_8 are closed Ω-structures, too. w_1–w_8 are all ϖ–structures, and $W(N) = \{w_1, \ldots, w_8\}$.

The following lemmas show that each A-circuit or closed Ω-structure can induce a siphon that may be emptied and hence lead to a deadlock. Their proofs can be referred to [39].

Lemma 1 [39]: Let θ be an A-circuit in APNS (N, M_0), and $M \in \mathbb{R}(N, M_0)$. Then $\phi(\theta) = R(\theta) \cup \{H(R(\theta)) \backslash \wp(\theta)\}$ is a siphon of N (induced by θ), and θ is saturated at M if and only if $\phi(\theta)$ is empty at M.

Lemma 2 [39]: Let $w = (\prod_1, \prod_2) = \pi_{11}\pi_{21}\ldots\pi_{1k}\pi_{2k}$ be a closed Ω-structure in APNS (N, M_0), and $M \in \mathbb{R}(N, M_0)$. Then $\phi(w) = R(\prod_1) \cup \{H(R(\prod_1)) \backslash \wp(\prod_1)\} \cup \wp(\prod_2)$ is a siphon of N (induced by w), and w is semi-saturated and semi-empty at M if and only if $\phi(w)$ is empty at M.

Let N be an APNS, $\aleph(N) = O(N) \cup W(N)$ be the set of maximal A-circuits and ϖ–structures, and $\mathrm{S}(N) = \{\phi(\alpha) \mid \alpha \in \aleph(N)\}$ be the set of siphons induced by A-circuits or ϖ–structures in $\aleph(N)$.

Example 3: For the APNS in Fig. 1, from Example 1, we know that $\theta_1 = p_{12}t_{13}p_{13}t_{14}$ is an A-circuit. By Lemma 1, the siphon induced by θ_1 is

$$\phi(\theta_1) = R(\theta_1) \cup \{H(R(\theta_1))\backslash\wp(\theta_1)\} = \{r_2, r_3, p_{14}\}$$

$w_1 = (\pi_1, \pi_2)$ is a closed Ω-structure. By Lemma 2, the siphon induced by w_1 is

$$\begin{aligned}\phi(w_1) &= R(\pi_1) \cup \{H(R(\pi_1))\backslash\wp(\pi_1)\} \cup \wp(\pi_2)\\ &= \{r_5, r_6, p_{22}, p_{42}, p_{43}\}.\end{aligned}$$

Let $M_0 = 5(p_{1s} + p_{2s} + p_{3s} + p_{4s}) + 2(r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_7)$ be the initial marking of N, and $M = p_{1s} + 5p_{2s} + p_{3s} + 5p_{4s} + 2(p_{12} + p_{13} + p_{31} + p_{32} + r_1 + r_4 + r_7) \in \mathbb{R}(N, M_0)$. $M(\wp(\theta_1)) = M_0(R(\theta_1)) = 4$. Hence θ_1 is saturated at M, and $M(\phi(\theta_1)) = 0$. $M(\wp(\pi_1)) = M_0(R(\pi_1)) = 4$ and $M(\wp(\pi_2)) = 0$, hence w_1 is semi-saturated and semi-empty at M, and $M(\phi(w_1)) = 0$.

Theorem 1 [39]: An APNS (N, M_0) is live if and only if none of A-circuits is saturated and none of closed Ω-structures is semi-saturated and semi-empty at $\forall M \in \mathbb{R}(N, M_0)$.

Its proof is given in [39]. Please refer to [39] for the details.

B. Deadlock Prevention Policies for APNSs

By Theorem 1, we know that only A-circuits and closed Ω-structures can lead an APNS to deadlocks. The system is

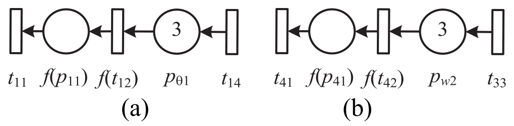

t_{11} $f(p_{11})$ $f(t_{12})$ $p_{\theta1}$ t_{14}
(a)
t_{41} $f(p_{41})$ $f(t_{42})$ p_{w2} t_{33}
(b)
3

Fig. 2. Controller for θ_1 and w_2 by [39].

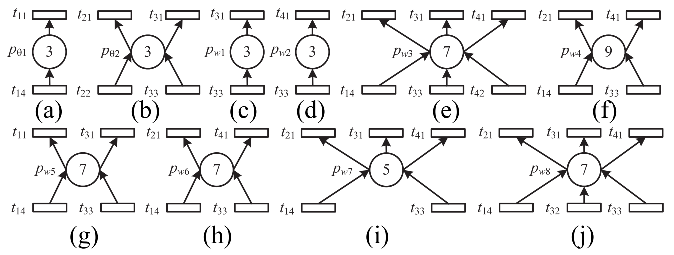

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)
$p_{\theta1}$ $p_{\theta2}$ p_{w1} p_{w2} p_{w3} p_{w4} p_{w5} p_{w6} p_{w7} p_{w8}

Fig. 3. PN controllers for maximal A-circuits and ϖ-structures.

live if and only if all siphons induced by A-circuits and closed Ω-structures are not emptied at any reachable marking. To avoid deadlock, it is necessary to guarantee that all siphons are not empty. The deadlock prevention policy in [39] is to add a PN controller to each A-circuit and closed Ω-structure such that their corresponding siphons cannot be emptied at any reachable marking. To make it easy to be integrated with a scheduling algorithm, we redefine it as follows.

Let π be an A-chain in APNS (N, M_0). A transition t is called an input or output transition of π if firing t increases or decreases the number of tokens in $\wp(\pi)$. Let $I(\pi)$ and $O(\pi)$ denote the sets of input and output transitions of π, respectively.

Let $T(\pi)$ denote the set of transitions in T_s from which there is an A-path to $I(\pi)$, i.e., $T(\pi) = \{t \in T_s \mid \text{An A-path from } t^{(a)} \text{ to } I(\pi) \text{ exists or } t \in I(\pi)\}$.

Definition 7: Let θ be an A-circuit in APNS (N, M_0). Define a weighted PN controller for θ

$$(C[\theta], M_\theta) = (P_\theta, T_\theta, F_\theta, W_\theta, M_\theta)$$

where $P_\theta = \{p_\theta\}$ and p_θ is the control place corresponding to θ with $M_\theta(p_\theta) = M_0(R(\theta)) - 1$, $T_\theta = T(\theta) \cup O(\theta)$, and $F_\theta = \{(p_\theta, t) \mid t \in T(\theta)\} \cup \{(t, p_\theta) \mid t \in O(\theta)\}$, and $W_\theta(t, p_\theta) = |{}^{(a)}t \cap \wp(\theta)| - |t^{(a)} \cap \wp(\theta)|$ for $(t, p_\theta) \in F_\theta$, and $W_\theta(p_\theta, t) = 1$ for $(p_\theta, t) \in F_\theta$.

Lemma 3: Let θ be an A-circuit in APNS (N, M_0), and $(C[\theta], M_\theta)$ be a PN controller for θ given in Definition 7. Then $(C[\theta], M_\theta)$ can avoid siphon $\phi(\theta)$ from being emptied. Let $(\underline{C}[\theta], \underline{M}_\theta)$ be the controller in [39] for θ. Then, $(C[\theta], M_\theta)$ is analogous to $(\underline{C}[\theta], \underline{M}_\theta)$.

Its proof is available at https://github.com/luojianchao/ILS4FAS.

Note that the controllers for A-circuits in this paper have some difference from those in [39]. Consider the APNS in Fig. 1. From Example 1, we know that $\theta_1 = p_{12}t_{13}p_{13}t_{14}$ is a maximal A-circuit. $T(\theta_1) = \{t_{11}\}$, and $O(\theta) = \{t_{14}\}$. The controller for θ_1 by Definition 7, $(C[\theta_1], M_{\theta1})$, is shown in Fig. 3(a), and that for θ_1 by [39], $(\underline{C}[\theta_1], \underline{M}_{\theta1})$, is shown in Fig. 2(a). Obviously, $(C[\theta_1], M_{\theta1})$ does not contain places and transitions in N^{-1} and hence is more concise than $(\underline{C}[\theta_1], \underline{M}_{\theta1})$. Although their structures are different, their function is the same. The function of $(C[\theta_1], M_{\theta1})$ is to constrain $M(p_{11}) + M(p_{12}) + M(p_{13}) + M(p_{\theta1}) = 3$, and $(\underline{C}[\theta_1], \underline{M}_{\theta1})$ is to constrain $M(p_{11}) + M(p_{12}) + M(p_{13}) + M(f(p_{11})) + M(p_{\theta1}) = 3$. Since $M(f(p_{11})) \geq 0$ and $M(p_{\theta1}) \geq 0$, $(C[\theta_1], M_{\theta1})$ and $(\underline{C}[\theta_1], \underline{M}_{\theta1})$ are both to constrain $M(p_{11}) + M(p_{12}) + M(p_{13}) \leq 3$. Hence, they are analogous.

Let $w = \pi_{11}\pi_{21}\ldots\pi_{1k}\pi_{2k} = (\prod_1, \prod_2)$ be a closed Ω-structure, where $\prod_1 = \{\pi_{1i}, i \in \mathbb{N}_k\}$ and $\prod_2 = \{\pi_{2i}, i \in \mathbb{N}_k\}$, and $\wp(\prod_1) = \{\wp(\pi_{1i}) \mid \pi_{1i} \in \prod_1\}$. A transition t is called an input or output of $\prod_1$ if firing t increases or decreases the number of tokens in $\wp(\prod_1)$. Let $I(\prod_1)$ and $O(\prod_1)$ denote the sets of input and output transitions of $\prod_1$, respectively.

Let $T(w)$ denote the set of transitions in T_s from which there is an A-path to $I(\prod_1)$, i.e., $T(w) = \{t \in T_s \mid \text{An A-path from } t^{(a)} \text{ to } I(\prod_1) \text{ exists or } t \in I(\prod_1)\}$.

Definition 8: Let $w = (\prod_1, \prod_2)$ be a closed Ω-structure in APNS (N, M_0), define a weighted PN controller for w

$$(C[w], \ M_w) = (P_w, T_w, F_w, W_w, M_w)$$

where $P_w = \{p_w\}$ and p_w is a control place corresponding to w, the initial marking is $M_w(p_w) = M_0(R(\prod_1)) - 1$, $T_w = T(w) \cup O(\prod_1)$, and $F_w = \{(p_w, t) \mid t \in T(w)\} \cup \{(t, p_w) \mid t \in O(w)\}$, and $W_w(t, p_w) = |{}^{(a)}t \cap \wp(\prod_1)| - |t^{(a)} \cap \wp(\prod_1)|$ for $(t, p_w) \in F_w$, and $W_w(p_w, t) = 1$ for $(p_w, t) \in F_w$.

Lemma 4: Let w be a closed Ω-structure in APNS (N, M_0), and $(C[w], M_w)$ be a PN controller for w given in Definition 8. Then $(C[w], M_w)$ can avoid siphon $\phi(w)$ from being emptied. Let $(\underline{C}[w], \underline{M}_w)$ be the controller in [39] for w. Then, $(C[w], M_w)$ is analogous to $(\underline{C}[w], \underline{M}_w)$.

Its proof is available at https://github.com/luojianchao/ILS4FAS.

Note that the controllers for closed Ω-structures in this paper have some difference from those in [39]. Consider the APNS in Fig. 1. From Example 1, we know that $w_2 = (\prod_1, \prod_2) = (\pi_2, \pi_1) = (\{p_{42}t_{43}p_{43}t_{33}\}, \{p_{31}t_{32}p_{32}t_{33}\})$ is a closed Ω-structure. $T(w_2) = \{t_{41}\}$, $O(\prod_1) = \{t_{33}\}$. The controller for w_2 by Definition 8, $(C[w_2], M_{w2})$, is shown in Fig. 3(d), and that for w_2 by [39], $(\underline{C}[w_2], \underline{M}_{w2})$, is shown in Fig. 2(b). Obviously, $(C[w_2], M_{w2})$ does not contain places and transitions in N^{-1} and hence is more concise than $(\underline{C}[w_2], \underline{M}_{w2})$. Although their structures are different, their function is the same. The function of $(C[w_2], M_{w2})$ is to ensure $M(p_{41}) + M(p_{42}) + M(p_{43}) + M(p_{w2}) = 3$, and $(\underline{C}[w_2], \underline{M}_{w2})$ is to ensure $M(p_{41}) + M(p_{42}) + M(p_{43}) + M(f(p_{41})) + M(p_{w2}) = 3$. Since $M(f(p_{41})) \geq 0$ and $M(p_{w2}) \geq 0$, $(C[w_2], M_{w2})$ and $(\underline{C}[w_2], \underline{M}_{w2})$ are both to ensure $M(p_{41}) + M(p_{42}) + M(p_{43}) \leq 3$. Hence, they are analogous.

Definition 9: A PN controller for APNS (N, M_0), (C, M_C), is the composition of all controllers for all maximal A-circuits and all ϖ–structures in $\aleph(N)$, i.e.,

$$(C, \ M_C) = \otimes_{\tau \in \aleph(N)}(C[\tau], \ M_\tau).$$

Then the controlled system can be modeled by the composition of (N, M_0) and (C, M_C) as follows. $(CN, M_{C0}) = (N, M_0) \otimes (C, M_C) = (P \cup P_s \cup P_e \cup P_R \cup P_C, T \cup T_C, F \cup F_C, M_{C0})$, where $M_{C0}(p) = M_0(p)$ $\forall p \in P \cup P_s \cup P_e \cup P_R$, and $M_{C0}(p) = M_C(p)$ $\forall p \in P_C$. When all activities of all parts are finished, CN reaches a final marking, denoted as M_{Cf}, which is defined as follows. $M_{Cf}(p) = 0\ \forall p \in P \cup P_s$, $M_{Cf}(p_{ie}) = M_{C0}(p_{is})\ \forall p_{ie} \in P_e$, $M_{Cf}(r_i) = C_i\ \forall r_i \in P_R$, and $M_{Cf}(p) = M_{C0}(p)\ \forall p \in P_C$.

Example 4: Consider the APNS N in Fig. 1, from Example 2, we know that $\aleph(N) = O(N) \cup W(N) = \{\theta_1, \theta_2, w_1, \ldots, w_8\}$. The controller for θ_1, θ_2, and w_1–w_8 are shown in Fig. 3(a)–(j), respectively.

Theorem 2: Let (N, M_0) be an APNS. Then the PN controller (C, M_C) for (N, M_0) in Definition 9 makes the controlled net deadlock-free.

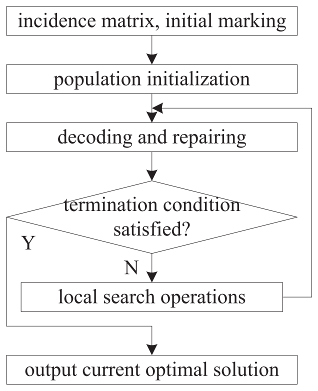
Its proof is available at https://github.com/luojianchao/ILS4FAS.

The controller in [39] contains some places and transitions in N^{-1}, but that in this paper does not. Thus, our controller is more concise. Besides, our controller has another advantage over that in [39], i.e., if it allows an activity and resource-enabled transition to fire at a making, then the transition can fire in the controlled system. However, the controller in [39] may allow such a firing only after some transitions in N^{-1} fire. Take the controllers for θ_1, i.e., $(C[\theta_1], M_{\theta 1})$ in Fig. 3(a) and $(\underline{C}[\theta_1], \underline{M}_{\theta 1})$ in Fig. 2(a) as an example. Both $(C[\theta_1], M_{\theta 1})$ and $(\underline{C}[\theta_1], \underline{M}_{\theta 1})$ allow t_{11} to fire in N at $M_0 = 5(p_{1s} + p_{2s} + p_{3s} + p_{4s}) + 2(r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_7)$. However, t_{11} is not enabled in $(\underline{C}[\theta_1], \underline{M}_{\theta 1}) \otimes (N, M_0)$ since $M(f(p_{11})) = 0$, and t_{11} is allowed to fire only after $f(t_{42})$ fires.

To illustrate the benefits of two above advantages, some notations are introduced. Let (C, M_C) and $(\underline{C}, \underline{M_C})$ denote our controller and the one in [39], respectively, $(CN, M_{C0}) = (C, M_C) \otimes (N, M_0)$, and $(\underline{CN}, \underline{M_{C0}}) = (\underline{C}, \underline{M_C}) \otimes (N, M_0)$. Usually, a deadlock-free scheduling algorithm integrates with a controller by taking it as a constraint for the original system. It means that a deadlock-free scheduling algorithm finds a schedule in the controlled net, e.g., (CN, M_{C0}) or $(\underline{CN}, \underline{M_{C0}})$. By the first advantage, we know that $(\underline{CN}, \underline{M_{C0}})$ has transitions and places in N^{-1}, while (CN, M_{C0}) does not. The search space of $(\underline{CN}, \underline{M_{C0}})$ is larger than that of (CN, M_{C0}). Thus, scheduling FMS with (C, M_C) can achieve higher search efficiency for an optimal schedule than doing it with $(\underline{C}, \underline{M_C})$. By the second advantage, we know that the schedule obtained in $(\underline{CN}, \underline{M_{C0}})$ cannot be executed in (N, M_0) directly since it may contain some transitions in N^{-1}, while the schedule obtained in (CN, M_{C0}) can be executed in (N, M_0) directly. Thus, scheduling the system with (C, M_C) is easier than doing so with $(\underline{C}, \underline{M_C})$.

IV. Deadlock-Free Local Search Algorithm for FASs

In order to obtain the optimal or suboptimal schedule to minimize the makespan and to avoid deadlock, a deadlock-free local search algorithm is proposed, which is an iterative algorithm in which the deadlock prevention policy is embedded. It is based on the following two main components with its flow chart in Fig. 4: 1) preliminaries of local search (coding, decoding, repairing, population generation, and termination condition) and 2) local search operations (perturbation, local search, and selection).



incidence matrix, initial marking
population initialization
decoding and repairing
termination condition satisfied?
Y
N
local search operations
output current optimal solution

Fig. 4. Flow chart of a deadlock-free local search scheduling algorithm.

A. Preliminaries of Local Search

1) Coding: In this paper, a solution for our scheduling problem is a sequence of the elements of a set of coded parts. The set of coded parts consists of a sequence of consecutive positive integers starting from 1, in which each integer corresponds to a part. Consider a system with n types of parts to be processed. The total number of parts to be processed is denoted as $\vartheta = \Psi_1 + \cdots + \Psi_n$, where Ψ_i $(i \in \mathbb{N}_n)$ is the number of type-i parts to be processed. Let $\vartheta_k = \sum_{j=1}^{k} \Psi_j$ denote the sum of first k $(k \in \mathbb{N}_n)$ types of parts to be processed. Specially, let $\vartheta_0 = 0$. Then, the set of type-i parts is coded as $\{\vartheta_{(i-1)} + 1, \vartheta_{(i-1)} + 2, \ldots, \vartheta_i\}$, and the set of all parts is coded as $\mathbb{N}_\vartheta$. Given a code $k \in \mathbb{N}_\vartheta$, let $\mu(k)$ denote the type of the part which is coded by k, i.e., $\mu(k) = i$ if $k \in \{\vartheta_{(i-1)} + 1, \vartheta_{(i-1)} + 2, \ldots, \vartheta_i\}$.

A *permutation* is defined as an order of the elements of a finite set. An order in which elements may appear more than once is called a *permutation with repetition*. A solution code of the scheduling problem is named as a *chromosome*. A chromosome is a permutation with repetition of elements in $\mathbb{N}_\vartheta$. Each code in a chromosome is named as a gene. For code $k \in \mathbb{N}_\vartheta$, it appears $(L_{\mu(k)} + 1)$ times in a chromosome, where $L_{\mu(k)}$ is the total number of manufacturing and assembly activities of a type-$\mu(k)$ part. The jth k represents the start of the part's jth activity, where $j \le L_{\mu(k)}$, and the last k represents the part's end activity. Different types of parts may share some identical activities. For simplicity, the codes of two identical activities are merged as one.

Example 5: Consider the APNS N in Fig. 1, and suppose that two type-1 parts and two type-2 parts are to be processed, i.e., $\Psi_1 = \Psi_2 = 2$. Then, $\vartheta_0 = 0$, $\vartheta_1 = \Psi_1 = 2$, $\vartheta_2 = \Psi_1 + \Psi_2 = 4$. The set of codes for type-1 parts is $\{\vartheta_0 + 1, \ldots, \vartheta_1\} = \{1, 2\}$, and that for type-2 parts is $\{\vartheta_1 + 1, \ldots, \vartheta_2\} = \{3, 4\}$. Thus, the set of all codes is $\mathbb{N}_4 = \{1, 2, 3, 4\}$. Since the fourth activity of a type-1 part and the third activity of a type-2 part are the same, their corresponding codes are merged as one. Similarly, the codes corresponding to end activities of a type-1 part and a type-2 part are merged as one. For simplicity, let the codes

Algorithm 1 RA

```
Input: chromosome $\Gamma$ and the controlled APNS $(CN, M_{C0}) = (N, M_0) \otimes (C, M_C)$;
Output: $\Gamma_1$; /* $\Gamma_1$ is the repaired chromosome */
 1: $M = M_{C0}$; /* initialization*/
 2: for all ($i = 0$; $i < |\Gamma|$ ; $i$++) do
 3:     let $j = 0$;
 4:     while ($\alpha(\Gamma)[i]$ is not enabled at $M$) do
 5:         $j$++;
 6:         swap($\Gamma[i]$, $\Gamma[i + j]$);
 7:     end while
 8:     $M[\alpha(\Gamma)[i]> M_1$;
 9:     $M = M_1$;
10: end for
11: $\Gamma_1 = \Gamma$;
12: return $\Gamma_1$;
```

of type-1 parts denote the merged ones. The total number of manufacturing and assembly activities of a type-1 parts is four, thus codes 1 and 2 appear $5 = 4 + 1$ times in a chromosome, where the first four codes 1 (or 2) represent the starts of a type-1 part's four activities, and the last one represents the part's end activity. The total number of manufacturing and assembly activities of a type-2 part is 3. Since the third and end activities of type-2 parts are denoted by the codes of type-1 parts, thus codes 3 and 4 appear $2 = 3 + 1 - 2$ times in a chromosome. Then, a chromosome can be expressed as $\Gamma_1 = (1, 2, 3, 1, 3, 2, 4, 2, 4, 1, 2, 1, 2, 1)$.

2) Decoding: In order to convert a chromosome into a sequence of codes that can be recognized by PN, the decoding technology is designed. Since a gene in a chromosome Γ represents a unique activity, Γ can be interpreted uniquely as a sequence of activities denoted as $o(\Gamma)$. By mapping each activity in $o(\Gamma)$ to its start transition, a chromosome Γ can be interpreted as a sequence of transitions, denoted as $\alpha(\Gamma)$.

Example 6: Consider chromosome Γ_1 in Example 5. The first 1 (or 2) in Γ_1 represents the start of a type-1 part's first activity, i.e., o_{11}, the second 1 (or 2) represents the start of a type-1 part's second activity, i.e., o_{12}, and the last 1 (or 2) represents a type-1 part's end activity, i.e., o_{1e}. Similarly, the first 3 (or 4) represents the start of a type-2 part's first activity, i.e., o_{21}, and the second 3 (or 4) represents for the start of a type-2 part's second activity, i.e., o_{22}. Thus, Γ_1 can be interpreted as $o(\Gamma_1) = (o_{11}, o_{11}, o_{21}, o_{12}, o_{22}, o_{12}, o_{21}, o_{13}, o_{22}, o_{13}, o_{14}, o_{14}, o_{1e}, o_{1e})$. By mapping each activity in $o(\Gamma)$ to its start transition, Γ can be interpreted as $\alpha(\Gamma_1) = (t_{11}, t_{11}, t_{21}, t_{12}, t_{22}, t_{12}, t_{21}, t_{13}, t_{22}, t_{13}, t_{14}, t_{14}, t_{15}, t_{15})$.

3) Repairing: The proposed repairing algorithm (RA) is a recursive procedure. Let (CN, M_{C0}) be a controlled APNS and Γ be a chromosome. At each step, select a gene in Γ, which can be interpreted as a transition that is enabled at the current marking M in CN, from unselected genes of Γ. This process corresponds to Lines 3–7 of RA. Then, update the current marking M by firing the transition corresponding to the selected gene from it. This process corresponds to Lines 8 and 9 of RA. Repeat this procedure till the set of unselected genes of $\Gamma = \varnothing$. This process corresponds to the for-loop in RA. We have the following result.

Theorem 3: RA is correct.

Its proof is available at https://github.com/luojianchao/ILS4FAS.

4) Initial Population Generation: A chromosome is generated in two steps: 1) generate a permutation with repetition of elements in N_ϑ, where $k \in \mathrm{N}_\vartheta$ appears $(L_{\mu(k)} + 1)$ times and 2) merge the codes in the permutation, which represent two identical activities of different types of parts as the one with smaller integer value.

5) Termination Condition: The maximum number of generations is used as a termination condition to stop the proposed algorithm.

B. Local Search Operations

Local search operations are used to create a child population from parent one. They include perturbation, local search, and selection.

1) Perturbation: This operation perturbs locally optimal solutions to escape from local optima and explore a new region of the search space. The EDA [18] is a population evolution algorithm based on a probability model, which owns strong global search ability. To take such advantage of EDA, a novel EDA-based perturbation policy (EDA-P) is presented.

Let L be the length of a chromosome, and β be the size of a population. In this paper, we choose $\varepsilon = \lceil \zeta_p \cdot \beta \rceil$ best chromosomes to establish the excellent population, where $\zeta_p \in (0, 1]$ is the parameter for EDA-P. The set of best chromosomes is denoted as $\{\Gamma_1, \ldots, \Gamma_\varepsilon\}$. The probability model $\mathbb{P}$ is an $L \times L$ matrix where $\mathbb{P}(i, j)$ is the probability that activity j is in the front of activity i. Formally, $\mathbb{P}(i, j) = (\sum_{k=1}^{\varepsilon} I_{ij}(\Gamma_k)) / (i\ .\ \varepsilon)$, where $I_{ij}(\Gamma_k)$ is a two-valued function defined as follows. If activity j is in or in the front of the ith position of a chromosome Γ_k, then $I_{ij}(\Gamma_k) = 1$; otherwise, $I_{ij}(\Gamma_k) = 0$.

A new chromosome Γ is generated based on $\mathbb{P}$ as follows. For the ith position of Γ, we select an activity by the roulette wheel method [25]. The selection probability of activity j is $\mathbb{P}(i, j)$. Once activity j is selected in position i, then it cannot be selected any more. So, we set $\mathbb{P}(k, j) = 0\ \forall k \in \mathbb{N}_L$ and $k > i$. Iteratively execute this procedure until a new chromosome is completed.

2) Local Search: This operation searches the neighbors of incumbent chromosomes to find better ones. Different local search operators have been proposed in [5] and [23]. They make contribution to the field in which they are used. However, they cannot be used in the problem studied in this paper. To take the size of a neighborhood into account, we define ζ_l to denote as the maximum ratio of the number of different genes between two chromosomes to L (the length of a chromosome). The number of neighbors of a chromosome increases with the value of ζ_l. According to [49], we know that the number of neighbors of an incumbent chromosome should not be too large. Otherwise, it is difficult to find any local optimum. When ζ_l is larger than 0.1, the number of neighbors of an incumbent chromosome is large enough. Thus, we roughly set the range of ζ_l as (0, 0.1]. A neighborhood chromosome is generated by randomly choosing $L \cdot \zeta_l$ genes from an incumbent chromosome, and then randomly putting them in the places where they are chosen.

Let χ be the maximum number of times of local search allowed to perform consecutively on an incumbent chromosome, which does not reduce its makespan. Given a chromosome Γ, let $\gamma(\Gamma) = \text{span}(\alpha(\Gamma))$ denote its makespan, and $\delta(\Gamma)$ record the number of times of consecutive local search on it which does not reduce its makespan. A local search operator starts from Γ. It iteratively performs the following process: find Γ_1 in the neighborhood of Γ; perform RA to repair Γ_1 to a feasible chromosome Γ_2; compare Γ_2 with Γ. If $\gamma(\Gamma_2)$ is less than $\gamma(\Gamma)$, Γ is replaced by Γ_2 and $\delta(\Gamma) = 0$; otherwise, $\delta(\Gamma) = \delta(\Gamma) + 1$. If $\delta(\Gamma)$ is equal to χ, local search is stopped. Generally speaking, the bigger χ is, the better the local search's performance. However, if χ is too big, the CPU time is too long. Thus, the effectiveness of χ is certain. In the following test, it is fixed as 10.

3) Selection: This operation selects a child population from incumbent population and newly generated population via local search. An acceptance criterion (AC) plays a crucial role in a local search algorithm [27]. It directly affects the intensification and diversification of a population. In order to keep high-quality and low-similarity chromosomes, we develop a novel Pareto dominance-based AC (PD-AC) [6]. Let Γ_b be the best chromosome found by now. A chromosome Γ is associated with two indicators $\gamma(\Gamma)$ and $S(\Gamma, \Gamma_b)$ representing the quality and similarity, respectively, where $S(\Gamma, \Gamma_b)$ is the number of positions in which Γ and Γ_b have the same genes.

A chromosome Γ is said to dominate another chromosome Γ_1, if $\gamma(\Gamma) \le \gamma(\Gamma_1)$ and $S(\Gamma, \Gamma_b) < S(\Gamma_1, \Gamma_b)$, or $\gamma(\Gamma) < \gamma(\Gamma_1)$ and $S(\Gamma, \Gamma_b) \le S(\Gamma_1, \Gamma_b)$. A chromosome is said to be nondominated if no other chromosome dominates it. The crowding distance of a chromosome Γ, denoted as $c(\Gamma)$, is defined as follows. If Γ is a boundary chromosome (chromosomes with smallest or largest indicators), then an infinite distance value is assigned to $c(\Gamma)$. Otherwise, $c(\Gamma)$ is assigned the sum of two values equal to the absolute normalized differences in the $\gamma(\Gamma)$ and $S(\Gamma, \Gamma_b)$ values of two adjacent chromosomes.

PD-AC is used to select child population from incumbent population and newly generated population by local search in three steps. First, PD-AC ranks each chromosome to be selected according to its dominance relationship. The nondominated solutions are in the first rank, the chromosomes that are only dominated by the nondominated chromosomes are in the second rank, and so on. Then, PD-AC computes the crowding distance of each chromosome. In the end, PD-AC selects β (the size of a population) chromosomes according to the rank and crowing distances by the following rule. PD-AC selects chromosomes with smaller rank. For the chromosomes in the same rank, the chromosome with biggest crowing distances is selected. The selecting process goes on until β chromosomes are selected. Since the current best solution is a nondominated solution with an infinite distance, it must be selected to the next generation.

TABLE I
PROCESSING TIMES OF ACTIVITIES OF THE APNS IN FIG. 1

$d(p_{11})$:27	$d(p_{21})$:29	$d(p_{31})$:31	$d(p_{41})$:25
$d(p_{12})$:16	$d(p_{22})$:18	$d(p_{32})$:23	$d(p_{42})$:19
$d(p_{13})$:34		$d(p_{33})$:24	$d(p_{43})$:38
$d(p_{14})$:22			

TABLE II
PARAMETER LEVELS

factors / levels	ζ_p	ζ_l	AC
1	0.2	0.02	RW-AC
2	0.4	0.05	B-AC
3	0.6	0.08	PD-AC

V. ILLUSTRATIVE EXAMPLE

The proposed deadlock-free local search algorithm is implemented in C++. The size of the code file is 26 KB. It is compiled by MSBuild 4.0 and run on a 1.7-GHz personal computer with 8GB RAM. The operating system of the computer is Windows 7 Ultimate. The APNS in Fig. 1 is used to test the performance of the proposed algorithm. Suppose that the numbers of type-1$\sim$4 parts are all 10. Let [1, 10], [11, 20], [21, 30], and [31, 40] denote the codes of type-1$\sim$4 parts, respectively. The processing time is randomly distributed in the range of [15, 40] for all manufacturing and assembly activities, as shown in Table I. The experimental data and main bodies of the code are available at https://github.com/luojianchao/ILS4FAS. The performance of the proposed deadlock-free local search algorithm may be affected by the size of the population, the maximum number of generations, ζ_p, ζ_l, AC, and perturbation policy. Generally speaking, the larger the population size or the maximum number of generations is, the better the algorithm's performance. Thus, the effectiveness of the population size and the maximum number of generations is certain. They are fixed as 20 and 2000, respectively. To test the effectiveness of the other factors, the following experiments are conducted.

In order to compare PD-AC with existing ACs, we implement another two ACs, i.e., roulette wheel-based AC (RW-AC) [25] and the better AC (B-AC) [49]. Let $\{\Gamma_j \mid j \in \mathbb{N}_{2\beta}\}$ be the set of all chromosomes in the incumbent population and newly generated one. In RW-AC, the selection probability of a chromosome Γ_j ($j \in \mathbb{N}_{2\beta}$) is $P(\Gamma_j) = f(\Gamma_j)/\sum_{i=1}^{2\beta} f(\Gamma_i)$, where $f(\Gamma_j) = (\max\{\gamma(\Gamma_k) \mid k \in \mathbb{N}_{2\beta}\} - \gamma(\Gamma_j) + 1)/(\max\{\gamma(\Gamma_k) \mid k \in \mathbb{N}_{2\beta}\} - \min\{\gamma(\Gamma_k) \mid k \in \mathbb{N}_{2\beta}\} + 1)$.

To test effectiveness of ζ_p, ζ_l, and AC, a design-of-experiment (DOE) [54] method is used. They all have three levels, as shown in Table II. Thus, a DOE with size $L_9(3^4)$ is selected. The proposed algorithm runs ten times at each factor combination. The average makespan is taken as an evaluation index. The final orthogonal experiment table is shown in Table III, and the range and rank of ζ_p, ζ_l, and AC are given in Table IV.

Since ζ_p owns the biggest range, it affects the performance of the proposed algorithm most. If it is too small, all chromosomes search toward the best chromosomes. The algorithm is easy to premature. If it is too big, the search direction is

TABLE III
ORTHOGONAL TABLE AND AVERAGE MAKESPAN

columns / experiments	ζ_p 1	ζ_l 2	AC 3	4	Average makespan
1	1	1	1	1	580.8
2	1	2	2	2	578.9
3	1	3	3	3	574.2
4	2	1	2	3	574.8
5	2	2	3	1	568.0
6	2	3	1	2	568.8
7	3	1	3	2	576.4
8	3	2	1	3	570.7
9	3	3	2	1	581.9

TABLE IV
RESPONSE TABLE

factors / levels	ζ_p	ζ_l	AC
1	1733.9	1732	1720.3
2	**1711.6**	**1717.6**	1735.6
3	1729	1724.9	**1718.6**
range	22.3	14.4	17
rank	1	3	2

TABLE V
SIMULATION RESULTS UNDER DIFFERENT PERTURBATION POLICY

EDA-P		R-P (0.1)		R-P (0.2)	
Makespan	RT(s)	Makespan	RT(s)	Makespan	RT(s)
570	81	558	83	590	82
585	79	578	76	581	78
572	76	573	75	588	71
562	79	561	78	580	81
575	74	586	79	587	76
585	73	538	77	563	79
598	79	567	81	573	74
578	78	575	78	596	76
558	75	566	79	574	73
575	79	578	78	547	77

too random. The algorithm performs like a random search. AC ranks second. Since PD-AC accepts chromosomes with both diversity and optimality, it can avoid repeated search to some extent, and thus achieves high search efficiency. Comparing with ζ_p and AC, ζ_l affects the performance of the algorithm least. However, a reasonable factor selection can help the algorithm perform better. Based on the previous analysis, they are selected as: $\zeta_p = 0.4$, $\zeta_l = 0.05$, and AC = PD-AC.

To test the effectiveness of EDA-P, a random perturbation (R-P) policy is developed. The R-P policy is the same as the local search operation proposed in the last section besides that the range of ζ_l is different. According to [49], we know that the strength of a perturbation policy should not be weaker than a local search. Otherwise, the algorithm can fall back into the local optimum just visited. Because the maximum ζ_l in the local search is set to be 0.1, let $\zeta_l \in [0.1, 1]$ in R-P. Two R-P policies with $\zeta_l = 0.1$ and 0.2, respectively, are implemented. The proposed algorithm under EDA-P ($\zeta_p = 0.4$), R-P (0.1), and R-P (0.2) all run ten times. Simulation results are shown in Table V, where RT means run time (in seconds).

The average makespan under EDA-P, R-P (0.1), and R-P (0.2) is 568, 575.8, and 577.9, respectively, and the average run time under EDA-P, R-P (0.1), and R-P (0.2) is 78.4, 77.3, and 76.7 s, respectively. The best solutions found under EDA-P, R-P (0.1), and R-P (0.2) have makespan 538, 558, and 547, respectively. Although the best solution found under R-P (0.2) is better than that under R-P (0.1), the average makespan under R-P (0.2) is larger than that under R-P (0.1). It means that the search under R-P (0.2) likes a random restart search. It may obtain some excellent solutions. However, the average performance of the found solutions is not so good. EDA-P performs the best among them. That means EDA-P owns stronger global search ability than both R-P (0.1) and R-P (0.2).

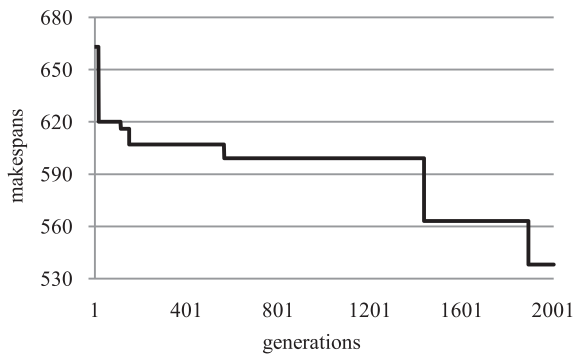


Fig. 5. Changing trend of the current best solution.

Our algorithm has two characteristics: 1) its optimization objective is to minimize the total completion time or makespan and 2) the solution with minimum makespan is kept for the next generation. Theoretically, if every solution can be searched, then it converges to the global optimum. Besides, we can analyze its convergence by the tested results. The process of finding the solution with makespan 538 is tracked in Fig. 5. From Fig. 5, we find that the makespan of the current best solution decreases with the number of generations. That means our algorithm converges toward the optimal solution as the search goes on. It is suitable for the studied NP-hard scheduling problem since it can generate better solutions as more time is given.

VI. CONCLUSION

This work studies the scheduling problem of deadlock-prone FASs. As far as we know, such a problem has never been reported in the existing literature. To solve it, a deadlock-free local search scheduling algorithm is proposed. It finds optimal or near-optimal deadlock-free schedules at given initial marking. To avoid deadlocks, an existing deadlock prevention policy is redefined in a concise way. The redefined one is analogous to the original one, but can be easily embedded in a scheduling algorithm. Then, the redefined deadlock prevention policy is embedded into an RA by which all chromosomes are amended into feasible ones that can be decoded into deadlock-free schedules. To enhance the global search ability of the proposed local search algorithm, a perturbation policy based on the EDA is proposed. Moreover, a novel PD-AC is developed to keep high-quality and high-diversity chromosomes. Experimental results show the effectiveness of the proposed scheduling algorithm. The proposed local search algorithm can be improved by proposing different local search operations. Improving deadlock prevention policies in terms of their optimality and computation complexity and their usage in scheduling requires future research.

Acknowledgment

The authors would like to thank the Editor, Associate Editor, and all anonymous reviewers for their thoughtful comments and suggestions that have greatly helped improve the presentation and technical quality of this paper.

References

[1] I. B. Abdallah, H. A. Elmaraghy, and T. Elmekkawy, "Deadlock-free scheduling in flexible manufacturing systems using Petri nets," *Int. J. Prod. Res.*, vol. 40, no. 12, pp. 2733–2756, Feb. 2002.

[2] O. T. Baruwa, M. A. Piera, and A. Guasch, "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 5, pp. 831–846, May 2015.

[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] S. F. Chew and M. A. Lawley, "Robust supervisory control for production systems with multiple resource failure," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 3, pp. 309–323, Jul. 2006.

[5] R. K. Congram, C. N. Potts, and S. L. Van De Velde, "An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem," *INFORMS J. Comput.*, vol. 14, no. 1, pp. 52–67, Jan. 2002.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[7] 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.

[8] M. P. Fanti, G. Maione, and B. Turchiano, "Design of supervisors to avoid deadlock in flexible assembly systems," *Int. J. Flexible Manuf. Syst.*, vol. 14, no. 2, pp. 157–175, Apr. 2002.

[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," *Inf. Sci.*, vol. 363, pp. 221–234, Oct. 2015.

[10] L. B. Han, K. Y. Xing, M. C. Zhou, X. Chen, and Z. X. Gao, "Efficient optimal deadlock control of flexible manufacturing systems," *IET Control Theory Appl.*, vol. 10, no. 10, pp. 1181–1186, Jun. 2016.

[11] L. B. Han, K. Y. Xing, X. Chen, H. Lei, and F. Wang, "Deadlock-free genetic scheduling for flexible manufacturing systems using Petri nets and deadlock controllers," *Int. J. Prod. Res.*, vol. 52, no. 5, pp. 1557–1572, Oct. 2013.

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

[13] 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.

[14] F.-S. Hsieh, "Robustness analysis of holonic assembly/disassembly processes with Petri nets," *Automatica*, vol. 44, no. 10, pp. 2538–2548, Oct. 2008.

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

[16] 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.

[17] Y. S. Huang, M. D. Jeng, X. L. Xie, and S. L. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 2, pp. 283–305, Jan. 2001.

[18] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer, 2001.

[19] Z. W. Li and M. C. Zhou, "Control of elementary and dependent siphons in Petri nets and their application," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 133–148, Jan. 2008.

[20] 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.

[21] J. C. Luo, K. Y. Xing, M. C. Zhou, X. L. Li, and X. N. Wang, "Deadlock-free scheduling of automated manufacturing systems using Petri nets and hybrid heuristic search," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 3, pp. 530–541, Mar. 2015.

[22] 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.

[23] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman, "A comparison of five heuristics for the multiple depot vehicle scheduling problem," *J. Sched.*, vol. 12, pp. 17–30, Feb. 2009.

[24] S. E. Ramaswamy and S. B. Joshi, "Deadlock-free schedules for automated manufacturing workstations," *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 391–400, Jun. 1996.

[25] C. R. Reeves and J. E. Rowe, *Genetic Algorithms-Principles and Perspectives: A Guide to GA Theory*. Norwell, MA, USA: Kluwer, 2003.

[26] 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.

[27] T. Stützle, "Iterated local search for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 174, no. 3, pp. 1519–1539, Nov. 2006.

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

[29] 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.

[30] 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.

[31] N. Q. Wu and M. C. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems based on a Petri net model," *IEEE Trans. Robot. Autom.*, vol. 17, no. 5, pp. 658–669, Oct. 2001.

[32] 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. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–10, Jan. 2013.

[33] N. Q. Wu and M. C. Zhou, "Real-time deadlock-free scheduling for semiconductor track systems based on colored timed Petri nets," *OR Spectr.*, vol. 29, no. 3, pp. 421–443, Jul. 2007.

[34] 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.

[35] N. Q. Wu, M. C. Zhou, F. Chu, and C. Chu, "A Petri-net-based scheduling strategy for dual-arm cluster tools with wafer revisiting," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1182–1194, Sep. 2013.

[36] Y. C. Wu, K. Y. Xing, J. C. Luo, and Y. X. Feng, "Robust deadlock control for automated manufacturing systems with an unreliable resource," *Inf. Sci.*, vols. 346–347, pp. 17–28, Jun. 2016.

[37] X. Gang and Z. M. Wu, "Deadlock-free scheduling strategy for automated production cell," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 113–122, Jan. 2004.

[38] 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. 289–296, Feb. 1996.

[39] 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.

[40] 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.

[41] K. Y. Xing, L. B. Han, M. C. Zhou, and F. Wang, "Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 603–615, Jun. 2012.

[42] H. Yue, K. Xing, and Z. Hu, "Robust supervisory control policy for avoiding deadlock in automated manufacturing systems with unreliable resources," *Int. J. Prod. Res.*, vol. 52, no. 6, pp. 1573–1591, Aug. 2013.

[43] M. C. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Singapore: World Sci., 1998.

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

[45] Y. Zhou, H. Hu, Y. Liu, and Z. Ding, "Collision and deadlock avoidance in multirobot systems: A distributed approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1712–1726, Jul. 2017.

[46] F. Lu, Q. Zeng, M. C. Zhou, Y. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2692262.

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

[48] Z. W. Li, M. C. Zhou, and N. Q. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 173–188, Mar. 2008.

[49] H. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Boston, MA, USA: Springer, 2010, pp. 363–397.

[50] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 1–14, Jan. 2015.

[51] P. Li *et al.*, "Iterated local search for distributed multiple assembly no-wait flowshop scheduling," in *Proc. IEEE Evol. Comput.*, San Sebastián, Spain, 2017, pp. 1565–1571.

[52] A. Subramanian, M. Battarra, and C. N. Potts, "An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 52, no. 9, pp. 2729–2742, Feb. 2014.

[53] J. Xu, C.-C. Wu, Y. Yin, and W.-C. Lin, "An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times," *Appl. Soft Comput.*, vol. 52, pp. 39–47, Mar. 2017.

[54] D. C. Montgomery, *Design and Analysis of Experiments*. New York, NY, USA: Wiley, 1984.

[55] X. Lu, M. C. Zhou, A. C. Ammari, and J. Ji, "Hybrid Petri nets for modeling and analysis of microgrid systems," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 4, pp. 349–356, Oct. 2016.

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

[57] 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. Automatica Sinica*, vol. 5, no. 1, pp. 270–280, Jan. 2018.

[58] G. Mitsuo and R. W. Cheng, *Genetic Algorithms and Engineering Optimization*. New York, NY, USA: Wiley, 2000.

[59] M. G. Filho, C. F. Barco, and R. F. T. Neto, "Using Genetic Algorithms to solve scheduling problems on flexible manufacturing systems (FMS): A literature survey, classification and analysis," *Flexible Services Manuf. J.*, vol. 26, no. 3, pp. 408–431, 2014.

[60] 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*, to be published, doi: 10.1002/asjc.1856.

[61] J. C. Luo, K. Y. Xing, M. C. Zhou, X. L. Li, and X. N. Wang, "Scheduling of deadlock and failure-prone automated manufacturing systems via hybrid heuristic search," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3283–3293, 2017.

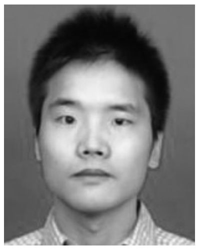
[62] F. J. Yang, N. Q. Wu, Y. Qao, M. C. Zhou, and Z. W. Li, "Scheduling of single-arm cluster tools for an atomic layer deposition process with residency time constraints," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 502–516, Mar. 2017.

[63] Y. Hou, N. Q. Wu, M. C. Zhou, and Z. W. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[64] J. C. Luo, Z. Q. Liu, and K. Y. Xing, "Hybrid branch and bound algorithms for the two-stage assembly scheduling problem with separated setup times," *Int. J. Prod. Res.*, to be published, doi: 10.1080/00207543.2018.1489156.

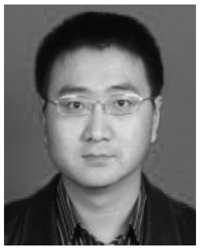
[65] C. Pan, M. C. Zhou, Y. Qiao, and N. Q. Wu, "Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 896–899, Apr. 2018.

[66] F. Yang, N. Wu, Y. Qiao, and M. C. Zhou, "Optimal one-wafer cyclic scheduling of time-constrained hybrid multicluster tools via Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 2920–2932, Nov. 2017.



JianChao Luo received the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2016.

He joined the Northwestern Polytechnical University, Xi'an, in 2017, where he is currently an Assistant Professor of Software Engineering. His current research interests include deadlock-free scheduling and control of AMS.



ZhiQiang Liu received the Ph.D. degree in computer science and engineering from Northwestern Polytechnical University, Xi'an, China, in 2007.

He is currently an Associate Professor of Software Engineering with Northwestern Polytechnical University. His current research interests include safety critical software, data analysis, and deadlock-free scheduling and control of AMSs.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and currently a Distinguished Professor of Electrical and Computer Engineering. He has over 700 publications including 12 books, over 400 journal papers (300+ in IEEE TRANSACTIONS), 11 patents, and 28 book-chapters. His current research interests include Petri nets, intelligent automation, Internet of Things, big data, Web services, and intelligent transportation.

Dr. Zhou is the Founding Editor of IEEE Press Book Series on Systems Science and Engineering. He is a fellow of the International Federation of Automatic Control, American Association for the Advancement of Science, and Chinese Association of Automation.

KeYi Xing received the Ph.D. degree in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 1994.

He is currently a Professor of Systems Engineering with the State Key Laboratory for Manufacturing Systems Engineering and the Systems Engineering Institute, Xi'an Jiaotong University. His current research interest includes control and scheduling of AMSs.