Deadlock-Free Scheduling of Automated Manufacturing Systems Using Petri Nets and Hybrid Heuristic Search

JianChao Luo, KeYi Xing, *Member, IEEE,* MengChu Zhou, *Fellow, IEEE,* XiaoLing Li, and XinNian Wang

***Abstract*—This paper focuses on the deadlock-free scheduling problem of automated manufacturing systems with shared resources and route flexibility, and develops novel scheduling methods by combining deadlock control policies and hybrid heuristic search. Place-timed Petri nets are used to model the systems and find a feasible sequence of firing transitions in the built model such that the firing time of its last transition is as small as possible. Based on the reachability graph of the net and a minimum processing time matrix, new heuristic and selection functions are designed to guide search processes. The proposed hybrid heuristic search is based on state space exploration and hence suffers from the state space explosion problem. In order to reduce the explored space, the search is restrained within a limited local search window. By embedding the deadlock-avoidance policies into the search processes, a novel deadlock-free hybrid heuristic search algorithm is developed. Experimental results indicate the effectiveness and superiority of the proposed algorithm over the state-of-the-art method.**

***Index Terms*—Automated manufacturing systems, deadlock-avoidance policy, heuristic search, Petri net, scheduling.**

I. Introduction

AN AUTOMATED manufacturing system (AMS) is a computer-controlled manufacturing system that consists of a finite set of resources and can process different kinds of parts based on a prescribed sequence of operations. Since there are high degrees of resource sharing in AMS, deadlocks may occur without proper control. In deadlock situations, the whole system or a part of it remains indefinitely blocked and cannot terminate its task. Thus, developing efficient control and scheduling algorithms to avoid deadlocks while optimizing the performance of systems is highly significant.

Manuscript received April 16, 2014; accepted July 20, 2014. Date of publication September 24, 2014; date of current version February 12, 2015. This work was supported by the National Natural Science Foundation of China under Grants 50975224 and 61034004. This paper was recommended by Associate Editor N. Wu.

J. Luo, K. Xing, X. Li, and X. Wang are with the State Key Laboratory for Manufacturing Systems Engineering, and Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: luo770826336@stu.xjtu.edu.cn; kyxing@sei.xjtu.edu.cn).

M. Zhou is with The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2014.2351375

A Petri net (PN) as a mathematical formalism and graphical tool is useful for modeling, analyzing and designing discrete event systems including AMSs. To cope with deadlocks in a PN formalism, researchers have presented three approaches: prevention, detection and recovery, and avoidance [2], [4], [9], [16], [20], [25], [28], [29], [37], [38]. Ezpeleta *et al*. [4] define a class of PNs called system of simple sequential processes with resources (S^3PR). They develop a deadlock control policy implemented by adding a control place and related arcs to each strict minimal siphon such that no siphon can become empty. Chu and Xie [2] develop an effective deadlock detection method for structurally bounded PNs by solving a mixed integer program (MIP). The proposed approach can check deadlock without the explicit enumeration of siphons. Based on their MIP method, Huang *et al*. [9] present an iterative deadlock prevention method for S^3PRs. It consists of two stages: siphon control and augmented siphon control. Based on the concepts of essential siphons and critical markings, Piroddi *et al*. [20] develop a selective siphon control method for S^3PR, which is highly permissive and with small size. Xing *et al*. [29] present the concept of a maximum prefect resource-transition circuit and use it to characterize deadlocks in S^3PRs. They derive an optimal polynomial complexity deadlock avoidance policy (DAP) for S^3PRs without a center-resource and a suboptimal polynomial complexity DAP for those with center-resources. In order to minimize the number of newly added places, Liu *et al*. [16] propose the concept of a transition cover. It is a set of maximal perfect resource-transition circuits (MPCs) in an S^3PR. They propose a deadlock prevention policy by adding a control place with a proper control variable to each MPC in an effective transition cover.

Traditional deadlock control methods can be used to guarantee the deadlock-free operation of AMS. Since they ignore the operation time, the operational performance of AMS is not guaranteed. Scheduling deadlock-prone AMSs to ensure their performance poses a new challenge.

Scheduling is one of the most important and difficult issues in the operation of AMSs. It belongs to the class of NP-hard combinatorial optimization problems, i.e., the optimal schedule for a large-scale problem is unlikely to be obtained within a reasonable time [24]. A strategy to combine PN simulation and AI-based systematic search within the PN

reachability graph (RG) is proposed in [7] for production scheduling. Lee and DiCesare [14] propose a heuristic search algorithm based on A^* search technique for scheduling a flexible manufacturing system (FMS) with routing flexibility, shared resources, lot size and concurrency. The algorithm can seek an optimal or near-optimal transition firing sequence in the RG of PNs. Sun *et al*. [23] propose a heuristic search based on the limited-expansion A algorithm and place-timed PNs for nondelay scheduling. The algorithm avoids the high memory requirement and exponential time with respect to the problem size by constraining the number of unexplored markings in a fixed range. In order to find a near-optimal or optimal solution for FMSs scheduling, Jeng and Chen [13] propose a heuristic function based on PN state equations. By pruning redundant markings to reduce the search space, their proposed heuristic algorithm can obtain near-optimal schedules within a reasonable computation time and smaller search space. Yu *et al*. [35] propose a heuristic function based on a resource cost reachability matrix. By integrating the heuristic function with two modified hybrid techniques based on the reduction of the scope of selection and recovery, respectively, they obtain some improvements over the previous work in [14]. New heuristic functions based on a lower bound reachability matrix and transition-timed PNs for the A^* algorithm are presented in [15]. The resulting algorithm is tested on random FMSs and yields satisfying results. Huang *et al*. [10] propose an admissible heuristic function based on place-timed PNs to schedule FMSs.

As search spaces grow exponentially with problem scales, to reduce the combinatorial explosion, Xiong and Zhou [30] combine the heuristic best-first strategy with the control backtracking based on the execution of place-timed PNs. Reyes *et al*. [22] propose a dynamic window search (DWS) algorithm to restrain the search in a limited local search window in which an policy is applied to select the most promising paths. Huang *et al*. [11] combine the A^* algorithm with the depth-first strategy based on PN execution. By invoking quicker termination conditions properly, the quality of their search result is controlled.

The scheduling problem of deadlock-prone AMSs involves not only the optimization of objective functions but also the handling of deadlock issues and therefore is widely recognized to be more difficult than that of deadlock-free ones [6]. Only a few research studies address it [1], [3], [5], [8], [21], [26], [27], [33]. Ramasway and Joshi [21] provide a mathematical model for the deadlock-free scheduling problem 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. Abdallaht *et al*. [1] use timed PNs to model AMSs and propose a deadlock-free scheduling algorithm. The algorithm is based on the depth-first search strategy together with a siphon truncation technique to minimize the mean flow time. Xu and Wu [5] develop a genetic algorithm based on PNs with infinite buffers and then analyze the deadlock that may occur in the scheduling and add some necessary buffers to avoid the deadlock. Yoon and Lee [33] propose a deadlock-free scheduling approach for photolithography equipment, which can perform scheduling and deadlock management in an efficient way in spite of failures of process modules. Dashora *et al*. [3] use extended colored timed PNs to model the dynamic behavior of AMSs and find a deadlock-free schedule with minimized makespan based on an evolutionary endosymbiotic learning automata algorithm. Wu and Zhou [26] solve the real-time deadlock-free scheduling problem of semiconductor track systems based on colored time resource-oriented PNs and in a hierarchical way. Heuristic rules are proposed with the help of a deadlock control policy to schedule the system in real time. Deadlock-free scheduling and control of cluster tools are conducted in [39], [40]. Xing *et al*. [27] embed a DAP into a genetic algorithm and develop a novel deadlock-free genetic algorithm for AMSs. Based on the deadlock search algorithm in [29], a one-step look-ahead method is developed to check the feasibility of a chromosome, and infeasible chromosomes are amended into feasible ones, which can be easily decoded into a feasible deadlock-free schedule. To improve the performance of deadlock-free genetic algorithm in [27], Han *et al*. [8] propose different kinds of crossover and mutation operations and perform them on chromosomes. Computational results via their method show the excellent improvement in performance.

This paper focuses on the scheduling problem for deadlock-prone AMSs. A schedule is obtained through a hybrid heuristic search in the extended reachability graph (ERG) of a PN. By combining DAPs with a DWS algorithm, a novel deadlock-free hybrid heuristic search algorithm is developed. In order to increase the chance of finding an optimal solution, three heuristic functions based on PNs and a minimum processing time matrix are proposed to select the promising paths. Moreover, two selection functions used to select enabled transitions at a vertex and determine whether a new vertex is kept or rejected are proposed. To reduce the search effort, DAPs are integrated in the search process to constrain the search to safe vertexes only. Among existing DAPs, to the authors' knowledge, only the DAP proposed in [29] is based on PNs and has optimality and polynomial complexity for our concerned AMSs. Thus, it is selected in this study. Our proposed hybrid heuristic search algorithm is suitable for large-scale deadlock-prone AMSs with routing flexibility, shared resources, lot size, and concurrency.

Section II introduces the PN modeling of AMSs. The PN-based DAPs with polynomial complexity are reviewed in Section III. In Section IV, three heuristic functions and two selection functions are defined. Based on them a deadlock-free hybrid heuristic search algorithm is presented by embedding DAPs in a DWS algorithm. The effectiveness of the proposed algorithm is demonstrated in Section V. Section VI concludes this paper.

II. Basic Definitions of PNS and Scheduling PN Models of AMSs

This section briefs the definitions and notations of PNs. More details can be found in [18] and [36]. Thereafter, the PN model of an AMS for scheduling is established.

A. Basic Definitions of PNs

A PN is a three-tuple $N = (P, T, F)$, where P and T are finite and disjoint sets, i.e., $P \cap T = \emptyset$, $P \neq \emptyset$, $T \neq \emptyset$. P is the set of places, and T is the set of transitions. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation or set of directed arcs.

Given a vertex $x \in P \cup T$, the preset of x is defined as ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$, and the postset is defined as $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$. These notations can be extended to a set, ${}^\bullet X = \cup_{x \in X} {}^\bullet x$ and $X^\bullet = \cup_{x \in X} x^\bullet$, where $X \subseteq P \cup T$. A path is a string $\alpha = x_1 x_2 \ldots x_k$, where $x_i \in P \cup T$ and $(x_i, x_{i+1}) \in F$, $i = 1, \ldots, k-1$. A path α is a circuit if $x_1 = x_k$.

A marking or state of N is a mapping $M : P \to Z$, where $Z = \{0, 1, 2, \ldots\}$. Given a place $p \in P$ and a marking M, $M(p)$ denotes the number of tokens in p at M. Let $S \subseteq P$ be a set of places. The sum of tokens in all places of S at M is denoted by $M(S)$, i.e., $M(S) = \Sigma_{p \in S} M(p)$. A PN 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 $\forall p \in {}^\bullet t$, $M(p) > 0$. An enabled transition t at M can be fired, generating a new reachable marking M', denoted as $M[t > M'$, where $M'(p) = M(p) - 1$, $\forall p \in {}^\bullet t \backslash t^\bullet$, $M'(p) = M(p) + 1$, $\forall p \in t^\bullet \backslash {}^\bullet t$, and otherwise, $M'(p) = M(p)$. A sequence of transitions $\alpha = t_1 t_2 \ldots t_k$ is feasible from a marking M if there exists $M_i[t_i > M_{i+1}$, $i = 1, 2, \ldots, k$, where $M_1 = M$. Let $R(N, M)$ denote the set of all reachable markings of N from M.

Let $R(N, M_0) = (V, E)$ denote the RG of (N, M_0), where V is the set of reachable markings of (N, M_0) and E is the set of directed arcs between reachable markings. From a reachable marking M_1 to another reachable marking M_2 there is an arc if there is a transition $t \in T$ such that $M_1[t > M_2$.

B. Place-Timed PN Scheduling Models of AMSs

The AMS studied in this paper consists of m types of resources and can process n types of parts. The set of resource types is marked as $R = \{r_i, i = 1, 2, \ldots, m\}$. Each type of resources may be a kind of machines, or robots. The capacity of r_i is a positive integer, denoted as $C(r_i)$, implying the maximum number of parts that such type of resources can simultaneously hold. Let r_{ij} denote the jth instance of r_i, where $j = 1, 2, \ldots, C(r_i)$. $ER = \{r_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, C(r_i)\}$ is the set of resource instances. The set of part types is marked as $Q = \{q, q = 1, 2, \ldots, n\}$. $\Psi(q)$ is the number of type-q parts to be processed.

A processing route of a part is a sequence of operations. A part may have more than one processing route and can select its route during the processing. Let $\Omega = \{w_i | 1 \leq i \leq |\Omega|\}$ denote the set of all processing routes in the system, and $\Omega(q) \subseteq \Omega$ denote the set of routes of type-q parts. Route w_i can be expressed as $w_i = o_{i1} o_{i2} \ldots o_{iL_i}$, where o_{ij} is the jth operation in w_i and L_i is the length of route w_i. Let o_{qs} and o_{qe} denote start and end operations of type-q parts, respectively. Then, the ith processing route of type-q parts with the additional operations can be denoted as $w_{q(i)} = o_{qs} o_{q(i)1} o_{q(i)2} \ldots o_{q(i)L_{q(i)}} o_{qe}$, where $w_{q(i)} \in \Omega(q)$. Different routes of a part may share some identical operations. For simplicity, identical operations in different routes are merged as one operation.

In our PN model, the ith processing route of type-q parts is modeled by a path $\alpha_{q(i)} = p_{qs} t_{q(i)1} p_{q(i)1} t_{q(i)2} p_{q(i)2} t_{q(i)3} \ldots p_{q(i)L_{q(i)}} t_{q(i)(L_{q(i)}+1)} p_{qe}$, where p_{qs} and p_{qe} represent the operations o_{qs} and o_{qe}, respectively, $p_{q(i)j}$ is an operation place representing operation $o_{q(i)j}$, $t_{q(i)j}$ represents the start of $o_{q(i)j}$, and $t_{q(i)(j+1)}$ represents the completion of $o_{q(i)j}$. Hence, the marked PN model of processing routes of type-q parts can be denoted as

$$N_q = \left(P_q \cup \{p_{qs}, p_{qe}\}, T_q, F_q, M_{q0}\right), q \in Q$$

where $P_q = \cup_{1 \leq i \leq |\Omega(q)|} \{p_{q(i)1}, \ldots, p_{q(i)L_{q(i)}}\}$, $T_q = \cup_{1 \leq i \leq |\Omega(q)|} \{t_{q(i)1}, t_{q(i)2}, \ldots, t_{q(i)(L_{q(i)}+1)}\}$, and $F_q = \cup_{1 \leq i \leq |\Omega(q)|} \{(p_{qs}, t_{q(i)1}), (t_{q(i)1}, p_{q(i)1}), \ldots, (p_{q(i)L_{q(i)}}, t_{q(i)(L_{q(i)}+1)}), (t_{q(i)(L_{q(i)}+1)}, p_{qe})\}$. M_{q0} is the initial marking, $M_{q0}(p) = 0$, $\forall p \in P_q \cup \{p_{qe}\}$, and $M_{q0}(p_{qs}) = \Psi(q)$. In N_q, $\forall t \in T_q$, $|{}^\bullet t| = |t^\bullet| = 1$. If $p \in P_q$, $|p^\bullet| > 1$, p is called a split place.

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

In this paper, suppose that each operation requires only one resource and two successive operations require different resources. Let $R(p)$ denote the resource required by operation place $p \in P_q$. 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:

$$(N, M_0) = \left(P \cup P_s \cup P_f \cup P_R, T, F, M_0\right)$$

where $P = \cup_{q \in Q} P_q$, $P_s = \{p_{qs} | q \in Q\}$, $P_f = \{p_{qe} | q \in Q\}$, $T = \cup_{q \in Q} T_q$, $F = F_Q \cup F_R$, and $F_Q = \cup_{q \in Q} F_q$. The initial marking M_0 is defined as $M_0(p_{qs}) = \Psi(q)$, $\forall p_{qs} \in P_s$, $M_0(p) = 0$, $\forall p \in P \cup P_f$, and $M_0(r) = C(r)$, $\forall r \in P_R$.

Let M_f be a marking of N such that $M_f(p_{qe}) = M_0(p_{qs})$, $\forall p_{qe} \in P_f$, $M_f(p) = 0, \forall p \in P \cup P_s$, and $M_f(r) = C(r)$, $\forall r \in P_R$. Call M_f the final marking of (N, M_0). A sequence of transitions τ is called a feasible schedule if $M_0[\tau > M_f$. For a marking $M \in R(N, M_0)$, M is safe if M_f is reachable from M; otherwise, M is unsafe.

For $p \in P$, let $d(p)$ denote the processing time of the operation modeled by p, and particularly, for $p \in P_s \cup P_f \cup P_R$, $d(p) = 0$. In the following, we refer to the above place-timed PN model as a PN for Scheduling (PNS). Let us illustrate the modeling methodology by using the following example.

Example 1: Consider an AMS with two types of resources r_1 and r_2, i.e., $R = \{r_1, r_2\}$. The system can process two types of parts, i.e., type-1 and type-2. Type-1 parts are processed first on r_1 and then on r_2. Type-2 parts are processed first on r_2 and then on r_1. The processing routes of type-1 and type-2 parts are $w_1 = o_{1s} o_{11} o_{12} o_{1e}$ and $w_2 = o_{2s} o_{21} o_{22} o_{2e}$, respectively. Then $\Omega = \{w_1, w_2\}$. The processing routes of type-1 and type-2 are modeled by $p_{1s} t_{11} p_{11} t_{12} p_{12} t_{13} p_{1e}$ and $p_{2s} t_{21} p_{21} t_{22} p_{22} t_{23} p_{2e}$, respectively. The model of the whole

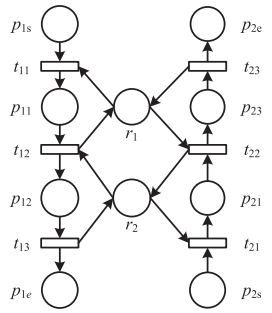

Fig. 1. PNS of the AMS in Example 1.

system is shown in Fig. 1, where the initial marking is not given.

Given $t \in T$, ${}^{(o)}t$ and $t^{(o)}$ denote its input and output operation places, and ${}^{(r)}t$ and $t^{(r)}$ denote its input and output resource places, respectively. Given $M \in R(N, M_0)$, t is operation-enabled at M if $M({}^{(o)}t) > 0$, t is resource-enabled at M if $M({}^{(r)}t) > 0$. In a PN, only transitions, which are operation and resource-enabled at the same time, can be fired. For a resource $r \in R$, let $H(r)$ denote the set of operation places which use resource r, i.e., $H(r) = \{p \in P | R(p) = r\}$.

III. Polynomial-Complexity DAPs for PNSs

According to the structures of PNS and S^{3}PR models [4] for AMSs, we know that DAPs for S^{3}PR are available for PNS of the same AMS. Hence, DAPs for S^{3}PRs proposed in [29] can be used for PNSs directly.

Xing *et al.* [29] use resource-transition circuits to characterize deadlock states in S^{3}PRs. Deadlock is characterized as a maximum prefect resource-transition circuit (MPRT-circuit) that is saturated at a reachable marking.

Definition 1: A directed circuit θ is a resource-transition circuit if it contains only resource places and transitions. Let $\zeta[\theta]$ and $\Re[\theta]$ denote the sets of all transitions and all resource places in θ, respectively. A resource-transition circuit θ is perfect if it satisfies $({}^{(o)}\zeta[\theta])^{\bullet} = \zeta[\theta]$. A perfect resource-transition circuit θ is said to be saturated under $M \in R(N, M_0)$, if $M({}^{(o)}\zeta[\theta]) = M_0(\Re[\theta])$. There is a unique MPRT-circuit with the resource set R_1, which contains all the resource-transition circuits with the same resource set R_1. Let θ_1 and θ_2 be two different MPRT-circuits. Every resource $r \in \Re[\theta_1] \cap \Re[\theta_2]$ with a capacity of one is called a ξ-resource.

Theorem 1 [29]: A marked S^{3}PR(N, M_0) is live if and only if no MPRT-circuit of N is saturated at any reachable marking of (N, M_0).

Example 2: Consider the PNS N in Fig. 1. There exists only one MPRT-circuit $\theta = t_{12}r_1t_{22}r_2t_{12}$. Let its initial marking be $M_0 = 2p_{1s} + p_{2s} + 2r_1 + r_2$. Then $M_1 = 2p_{11} + p_{21}$ is a reachable marking. At M_1, θ is saturated, i.e., $M_1({}^{(o)}\zeta[\theta]) = 3 = M_0(\Re[\theta])$ and all transitions are dead.

Based on the deadlock characterization, Xing *et al.* [29] derive an optimal polynomial-complexity DAP for S^{3}PRs without ξ-resources by a one-step look-ahead method. Correspondingly, a deadlock search (DS) algorithm, which prohibits S^{3}PRs from safe markings to deadlock ones, is proposed. For an S^{3}PR with ξ-resources, by reducing them and using the optimal DAP of the reduced net without ξ-resources, a suboptimal DAP is synthesized, and its computation is also of polynomial complexity [29].

IV. Deadlock-Free Scheduling via Hybrid Heuristic Search

The behavior of an AMS can be completely tracked by the RG of its PNS. Hence, an AMS scheduling problem can be transformed into a search problem in the RG of its PNS, and a schedule is a sequence of all transitions in a path from M_0 to M_f.

To solve the scheduling problem of AMSs, Lee and DiCesare [14] combine the A^* algorithm with searches within the RG of PN and develop a new scheduling method. A^* tries to reduce the search space by selecting the promising markings based on a heuristic function $f(M) = g(M) + h(M)$, where M is the current marking reachable from M_0, $g(M)$ is the current shortest time determined so far from M_0 to M, and $h(M)$ is an estimate of the remaining time from M to the final marking M_f along an optimal path which goes through M. To guarantee that A^* finds an optimal solution, $h(M)$ must be admissible [19], i.e., $h(M) \leq h^*(M)$ holds for any reachable marking M, where $h^*(M)$ is the actual minimum time of routes going from M to M_f.

As the A^* algorithm explores the search space in a best-first manner, it may lead to a large number of markings being explored before finding the first solution [19]. To reduce the search space, different heuristic functions are proposed and used [13], [14], [23], and [32]. They make A^* prefer deeper markings in an RG. Such an algorithm is a depth-first search which, unfortunately, often produces poor quality solutions.

In order to reduce the size of the search space in RG while to increase the chance of finding a better suboptimal solution, Reyes *et al.* [22] propose a DWS. It follows the basic schema of A^* and aims at reducing the space to be explored by: 1) constraining the current search space to a window and 2) choosing the most promising markings for further exploration through a heuristic function. Motivated by their work we use the ERG of a PNS, which is defined next, to develop three heuristic functions and find solutions.

A. Extended Reachability Graph (ERG) of PNS

For a reachable marking M of PNS(N, M_0), there are many feasible sequences of transitions that can lead the net from M_0 to M. This means that reaching the same marking could have different routes and hence, at different time. In order to use the information of transition sequences in heuristic search in the RG of a PNS, this paper extends the RG by combining a reachable marking M with its sequence of firing transitions α to form a vertex (M, α). In other words, in the ERG of PNS(N, M_0), a vertex contains not only a marking, but also its transition sequence to reach the marking.

Let $\boldsymbol{R}(N, M_0) = (V, E)$ denote the ERG of PNS(N, M_0), where V is the set of vertexes and E is the set of directed arcs.

A vertex is a combination of a marking M and its corresponding feasible sequence of firing transitions α, denoted as (M, α), such that $M_0[\alpha > M$. Let (M_1, α_1) and (M_2, α_2) be two vertexes. (M_1, α_1) and (M_2, α_2) are the same if $M_1 = M_2$ and $\alpha_1 = \alpha_2$. There is an arc from (M_1, α_1) to (M_2, α_2) if there is a transition $t \in T$ such that $M_1[t > M_2$ and $\alpha_2 = \alpha_1 t$.

Example 3: Consider the PNS shown in Fig. 1. Let $M_0 = p_{1s} + p_{2s} + 2r_1 + r_2$ be the initial marking. The processing time of parts in places p_{11}, p_{12}, p_{21}, and p_{22} is 30, 10, 10, and 25, respectively. There are two sequences of firing transitions $\alpha_1 = t_{21}t_{22}t_{11}t_{23}t_{12}$ and $\alpha_2 = t_{21}t_{11}t_{22}t_{12}t_{23}$, which can lead to the same marking $M = p_{12} + p_{2e} + 2r_1$ from M_0. The firing time of last transitions in α_1 and α_2 is different which is 40 and 35, respectively. Thus, in this paper we use two vertexes (M, α_1) and (M, α_2) in the ERG to indicate two different paths.

B. New Heuristic Functions

To solve the scheduling problem of AMSs, many heuristic functions based on the RG have been proposed [10], [12]–[14], [17], [22], [23], [30]–[32], [34], and all of them use reachable markings as the variable or argument and have the expression $f(M) = g(M) + h(M)$. In this paper, based on an ERG, three new heuristic functions are proposed. They have the form $f(M, \alpha) = g(M, \alpha) + h(M, \alpha)$ with $h(M, \alpha)$ varying while $g(M, \alpha)$ being defined as the firing time of the last transition in α. $h(M, \alpha)$ is based on the minimum processing time (MPT) matrix defined as follows.

Definition 2: Let $(N, M_0) = (P \cup P_s \cup P_f \cup P_R, T, F, M_0)$ be a PNS. Let p and p' be two operation places in (N, M_0) and $\alpha = pt_1p_1t_2 \ldots t_np_n(= p')$ be an operation path from p to p', where an operation path means that its all places are operation ones. Let $D(\alpha) = \sum_{i=1,\ldots,n} d(p_i)$. Define the minimal processing time from p to p' as $X(p, p') = \min\{D(\alpha) | \alpha$ is an operation path from p to $p'\}$. For convenience, if there is no operation path from p to p', define $X(p, p') = \infty$. Then $X(p, p')$ can be regarded as a square matrix indexed by operation places, called as an MPT matrix.

Since for every place p, there is a unique end operation place $p_f \in P_f$, let $X(p)$ denote $X(p, p_f)$ for simplicity.

Definition 3: Let $M_0[\alpha > M$, p be an operation place, and $M(p) > 0$. Let $j(p, i)$ denote the ith part or token in place p at M. For vertex (M, α), let $RT(j(p, i), M, \alpha)$ denote the remaining processing or sojourn time of part $j(p, i)$ in p, i.e., if $j(p, i)$ enters p at time $g(M_1, \alpha_1)$, then $RT(j(p, i), M, \alpha) = \max\{0, d(p) - (g(M, \alpha) - g(M_1, \alpha_1))\}$. Define

$$h_1(M, \alpha) = \sum_{j(p,i) \in J} (RT(j(p, i), M, \alpha) + X(p)) / |ER|.$$

Then $h_1(M, \alpha)|ER|$ is the sum of minimum processing time needed to finish all operations of parts from (M, α) under the assumption that (A1) all resources are available and parts do not wait for resources, and (A2) all parts are processed along the operation path with minimum processing time. Hence $h_1(M, \alpha)$ is the average minimum time in each resource instance needed to finish all the parts from (M, α) under assumptions (A1) and (A2).

A resource instance may have two statuses: idle and busy (it is processing a part). In order to estimate remaining sojourn time of parts in the system accurately, it is necessary to take the idle time of resources into consideration. For this purpose, we need the following concepts.

Let $OT(t)$ and $RT(t)$ denote the earliest possible time to operation and resource-enable transition t from $g(M, \alpha)$, respectively. They can be computed as follows.

Let $p = {}^{(o)}t$. If $M(p) > 0$, then $OT(t) = \min\{RT(j(p, i), M, \alpha) | i = 1, \ldots, M(p)\} + g(M, \alpha)$; if $M(p) = 0$, let V be the set of all places $q \in P \cup P_s$ such that from q to p there is an operation path $\beta = qt_1q_1t_2 \ldots q_{k-1}t_kp$, $M(q) > 0$ and $M(q_i) = 0$, $i = 1, 2, \ldots, k-1$. Then if $V \neq \emptyset$, $OT(t) = \min_{q \in V}\{\min\{RT(j(q, i), M, \alpha) | i = 1, \ldots, M(q)\} + X(q, p)\} + g(M, \alpha)$; otherwise $OT(t) = \infty$.

Let $t \in T$, $r = {}^{(r)}t$. If $M(r) > 0$, then $RT(t) = g(M, \alpha)$; otherwise, let J_1 be the set of parts occupying r at M, i.e., $J_1 = \{j(p, i) | p \in H(r), i = 1, \ldots, M(p)\}$ and $l(j(p, i))$ be the earliest possible time for $j(p, i) \in J_1$ leaving place p. Then $RT(t) = \min\{l(j(p, i)) | j(p, i) \in J_1\}$.

It is easy to know that $\forall t_1, t_2 \in r^\bullet$, $RT(t_1) = RT(t_2)$.

Let $G(r, M, \alpha) = \min\{\max\{OT(t), RT(t)\} - RT(t) | t \in r^\bullet\}$. Then $G(r, M, \alpha)$ denotes the minimum idle time of r before being used from $g(M, \alpha)$ to when r is used, that is, the interval time between the earliest possible enabled time and the earliest possible resource-enabled time of some transition in $r^\bullet$. Now, we can define $h_2(M, \alpha)$.

Definition 4: Let $M \in R(N, M_0)$ and $M_0[\alpha > M$. Define

$$h_2(M, \alpha) = \left(\sum_{j(p,i) \in J} (RT(j(p, i), M, \alpha) + X(p)) + \sum_{r \in R} \delta(r, M, \alpha) G(r, M, \alpha) \right) \Big/ |ER|$$

where $\delta(r, M, \alpha)$ is a two-valued function, which takes the value of either 0 or 1. If there is $p \in {}^{(o)}(r^\bullet)$ such that $M(p) > 0$ and $G(r, M, \alpha) = \min\{G(r', (M, \alpha)) | r' \in {}^{(r)}(p^\bullet)\}$, then $\delta(r, M, \alpha) = 1$, otherwise, $\delta(r, M, \alpha) = 0$.

Similarly to $h_1(M, \alpha)$, $h_2(M, \alpha)$ is computed under assumptions (A1) and (A2) and $h_2(M, \alpha)|ER|$ is the sum of minimum processing time needed to finish all operations of parts from (M, α) and the minimum idle time of all resources before being used newly.

Next, we prove that both h_1 and h_2 are admissible. To this end, let $r \in ER$, σ^* be the actual optimal part processing sequence of firing transitions from (M, α) to M_f, and $d_1(M, \alpha, \sigma^*, r)$ and $d_2(M, \alpha, \sigma^*, r)$ be the sums of total idle time and processing time of r from $g(M, \alpha)$ to the firing time of the last transition in σ^*, denoted as $l(\sigma^*)$, respectively. Then for any two resources r and r' in ER, we have $d_1(M, \alpha, \sigma^*, r) + d_2(M, \alpha, \sigma^*, r) = d_1(M, \alpha, \sigma^*, r') + d_2(M, \alpha, \sigma^*, r') = l(\sigma^*) - g(M, \alpha)$. Let $h^*(M, \alpha) = d_1(M, \alpha, \sigma^*, r) + d_2(M, \alpha, \sigma^*, r)$. Then $h^*(M, \alpha)$ can be expressed as $h^*(M, \alpha) = (\sum_{r \in ER} d_1(M, \alpha, \sigma^*, r) + \sum_{r \in ER} d_2(M, \alpha, \sigma^*, r))/|ER|$.

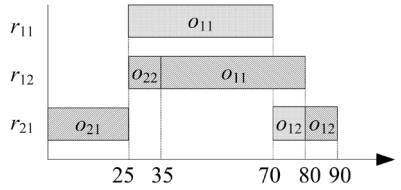

Fig. 2. Gantt chart of schedule $\alpha_1\alpha_2$.

Theorem 2: Heuristic functions in Definitions 3 and 4 are admissible, i.e., $h_1(M,\alpha) \le h^*(M,\alpha)$ and $h_2(M,\alpha) \le h^*(M,\alpha)$.

The proof is given in the Appendix.

Let h and h' be two heuristic functions. h is said to be more informed than h' if 1) both h and h' are admissible and 2) $h(M,\alpha) \ge h'(M,\alpha)$ for any node (M,α).

Theorem 3: h_2 is more informed than h_1.

Proof: By Theorem 2, h_2 and h_1 both are admissible. By the proof of Theorem 2, $h_2(M,\alpha) \ge h_1(M,\alpha)$ holds for any (M,α). Thus, h_2 is more informed than h_1. ■

Definition 5: Let $M \in R(N,M_0)$, and $M_0[\alpha > M$. Define

$$h_3(M,\alpha) = \left(\sum_{j(p,i)\in J} (RT(j(p,i),M,\alpha) + X(p)) + \sum_{r\in R} K(r,M,\alpha)G(r,M,\alpha) \right) \Big/ |ER|$$

where $K(r,M,\alpha)$ is the number of new parts to be arranged on r in the remaining processing from (M,α) to (M_f,α') by arranging all the parts along their shortest operation paths.

Note that $\sum_{j(p,i)\in J}(RT(j(p,i),M,\alpha) + X(p))$ is the minimum total processing time needed to finish all parts from $g(M,\alpha)$, and $\sum_{r\in R} K(r,M,\alpha)G(r,M,\alpha)$ is an estimate of the total idle time. In $h_3(M,\alpha)$, all idle time of resources before being used from $g(M,\alpha)$ is considered.

We next show that $h_3(M,\alpha)$ is not admissible by the following example. Although it is not, it can be used to find solutions near the optimal one, to be shown later in this paper. Let us illustrate the proposed heuristic functions by using the following example.

Example 4: For the PNS shown in Fig. 1. Let $M_0 = 2p_{1s}+p_{2s}+2r_1+r_2$ be the initial marking, and the processing time of parts in places p_{11}, p_{12}, p_{21}, and p_{22} be 45, 10, 25, and 10, respectively. Consider a sequence of firing transitions $\alpha_1 = t_{21}t_{22}t_{11}t_{23}$. $M_0[\alpha_1 > M = p_{1s}+p_{11}+p_{2e}+r_1+r_2$. Then $g(M,\alpha_1)$, which is the firing time of t_{23}, is 35. At $g(M,\alpha_1)$, we can work out an optimal remaining sequence of firing transitions from $g(M,\alpha_1)$, i.e., $\alpha_2 = t_{11}t_{12}t_{13}t_{12}t_{13}$. The Gantt chart of $\alpha_1\alpha_2$ is shown in Fig. 2 and from it, we know $h^*(M,\ \alpha_1) = 55$.

At (M,α_1), either of p_{1s} and p_{11} has a part, $j(p_{1s},1)$ and $j(p_{11},1)$ and their remaining time in p_{1s} and p_{11} is $RT(j(p_{1s},1),M,\alpha_1) = \max\{0, 0-(35-0)\} = 0$ and $RT(j(p_{11},1),M,\alpha_1) = \max\{0, 45-(35-25)\} = 35$, respectively. The shortest paths from p_{1s} and p_{11} to p_{1e} are $p_{1s}t_{11}p_{11}t_{12}p_{12}t_{13}p_{1e}$ and $p_{11}t_{12}p_{12}t_{13}p_{1e}$, respectively. By Definition 2, $X(p_{1s}) = d(p_{11})+d(p_{12})+d(p_{1e}) = 55$, and $X(p_{11}) = d(p_{12})+d(p_{1e}) = 10$. Finally, $h_1(M,\alpha_1) = ((RT(j(p_{1s},1),M,\alpha_1)+X(p_{1s}))+(RT(j(p_{11},1),M,\alpha_1)+X(p_{11})))/|ER| = ((0+55)+(35+10))/3 = 100/3$.

Since ${}^{(o)}t_{11} = p_{1s}$ and $M(p_{1s}) = 1$, $OT(t_{11}) = RT(j(p_{1s},1),M,\alpha_1)+g(M,\alpha_1) = 0+35 = 35$. Similarly, $OT(t_{12}) = RT(j(p_{11},1),M,\alpha_1)+g(M,\alpha_1) = 35+35 = 70$. Since $M(p_{1s}) = 0$, $OT(t_{21}) = \infty$. Similarly, $OT(t_{22}) = \infty$.

Since ${}^{(r)}t_{11} = {}^{(r)}t_{22} = r_1$ and $M(r_1) = 1$ at $g(M,\alpha_1)$, $RT(t_{11}) = RT(t_{22}) = g(M,\alpha_1) = 35$. Similarly, $RT(t_{12}) = RT(t_{21}) = g(M,\alpha_1) = 35$.

Since $r_1^\bullet = \{t_{11},t_{22}\}$, $G(r_1,M,\alpha_1) = \min\{\max\{OT(t), RT(t)\} - RT(t) | t \in r_1^\bullet\} = \min\{\max\{35,35\}-35, \max\{\infty,35\}-35\} = 0$. $r_2^\bullet = \{t_{12},t_{21}\}$, $G(r_2,M,\alpha_1) = \min\{\max\{OT(t),RT(t)\} - RT(t) | t \in r_2^\bullet\} = \min\{\max\{70,35\}-35, \max\{\infty,35\}-35\} = 35$.

Since $p_{1s} \in {}^{(o)}(r_1^\bullet)$, $M(p_{1s}) = 1$, and $|{}^{(r)}(p_{1s}^\bullet)| = 1$, we have $\delta(r_1,M,\alpha_1) = 1$. Similarly, $\delta(r_2,M,\alpha_1) = 1$. Then $h_2(M,\alpha_1) = (((RT(j(p_{1s},1),M,\alpha_1)+X(p_{1s}))+(RT(j(p_{11},1),M,\alpha_1)+X(p_{11})))+(\delta(r_1,M,\alpha_1)G(r_1,M,\alpha_1)+\delta(r_2,M,\alpha_1)G(r_2,M,\alpha_1)))/|ER| = (((0+55)+(35+10))+(0+35))/3 = 45$. Obviously, $h_1(M,\alpha_1) = 100/3$ and $h_2(M,\alpha_1) = 45$. Both are less than $h^*(M,\alpha_1) = 55$, while $h_2(M,\alpha_1)$ is larger than $h_1(M,\alpha_1)$.

By arranging parts $j(p_{1s},1)$ and $j(p_{11},1)$ along their shortest operation paths, $p_{1s}t_{11}p_{11}t_{12}p_{12}t_{13}p_{1e}$ and $p_{11}t_{12}p_{12}t_{13}p_{1e}$, respectively, there will be a part to be arranged on r_1 and two parts on r_2, i.e., $K(r_1,M,\alpha_1) = 1$ and $K(r_2,M,\alpha_1) = 2$. According to Definition 5, $h_3(M,\alpha_1) = (((RT(j(p_{1s},1),M,\alpha_1)+X(p_{1s}))+(RT(j(p_{11},1),M,\alpha_1)+X(p_{11})))+(K(r_1,M,\alpha_1)G(r_1,M,\alpha_1)+K(r_2,M,\alpha_1)G(r_2,M,\alpha_1)))/|ER| = (((0+55)+(35+10))+(0+2*35))/3 = 56.67 > 55 = h^*(M,\alpha_1)$, which means that $h_3(M,\alpha_1)$ is not admissible.

C. DWS

The application of A^* is only affordable for small-scale problems. Consequently, for any sizable ones, the goal of optimality must be sacrificed in favor of computation complexity reduction. By constraining the search space in a dynamic window, DWS reduces the size of the RG to be explored [17], [22]. In order to prevent the rapid growth of the number of explored vertexes, the size of the window is limited; while in order to increase the chance of finding an optimal solution, we propose two selection functions to choose the promising transitions and determine whether a new vertex is kept or rejected.

1) Window Parameters and Moving Rules: The depth of a vertex (M,α), denoted as $depth(M,\alpha)$, is the number of transitions in α. Let *bottom_depth* and *top_depth* denote the minimum and maximum depths of the vertexes which can be contained in the current search window, respectively. The first decision variable *high* is introduced to determine the distance between *bottom_depth* and *top_depth*. The search window only advances down vertically, and the distance between *bottom_depth* and *top_depth* is fixed to *high*.

All vertexes in the current search window are divided into two classes: explored and unexplored. Let $\chi_u(l)$ and $\chi_e(l)$ be the numbers of unexplored and explored vertexes at depth l, respectively. The search window explores the unexplored vertexes repeatedly starting from the root vertex (M_0, ε), where ε is an empty string. In each repeating process, an unexplored vertex (M, α) is selected according to certain rules and then explored and marked as explored. In exploring vertex (M, α), some enabled transitions at M are selected and fired, thus creating some new unexplored vertexes. At the same time, the search window advances when some conditions are met.

Now we introduce three other decision variables: max _ *size*, max _*vertexes*, and max _*top*, which stand for the maximum number of vertexes which can be selected to be explored at each depth, the maximum number of successor vertexes to be considered for each vertex, and the maximum number of vertexes which can be contained at *top_depth*, respectively.

In the current window, if $top\_depth < depth(M_f, \alpha)$ and the search window remains static, i.e., both *bottom_depth* and *top_depth* remain unchanged, then vertexes with *top_depth* can be created but cannot be explored. However, while $top\_depth < depth(M_f, \alpha)$, it is necessary to advance the search window to deeper vertexes in the ERG in order to reach a final vertex. Two rules used in the advance are proposed in [17].

Rule 1: if $(\chi_u(top\_depth) \geq \max\_top)$ {
discard all vertexes at level *bottom_depth*;
$top\_depth = top\_depth + 1$;
$bottom\_depth = bottom\_depth + 1$;}

Rule 2: if $(\chi_u(bottom\_depth) = 0)$ {
$top\_depth = top\_depth + 1$;
$bottom\_depth = bottom\_depth + 1$;}.

For the current search window, let $BEST(l)$ be the minimum value of f at all vertexes at depth l. max _*size* is used to restrict the number of vertexes at each depth. Once $\chi_e(l) \geq \max\_size$, the newly generated vertex (M, α) with depth l will not be included into the current window unless $f(M, \alpha) < BEST(l)$, that is, the value of f at vertex (M, α) is smaller than that at any other vertex at the same depth. This can be represented by the following newly proposed rule.

Rule 3: Let (M, α) be a newly generated vertex in the current window;
if $((\chi_e(depth(M, \alpha)) < \max\_size)$
or $(f(M, \alpha) < BEST(depth(M, \alpha))))$ {
insert (M, α) into the current window;}

2) Selection Functions: To increase the chance of finding the optimal solution, two selection functions are proposed in this paper. The first one is used to select enabled transitions at a vertex, while the second one is used to determine whether a new vertex is kept or not.

Let $M \in R(N, M_0)$, $M_0[\alpha > M$, t be a transition enabled at (M, α), $M[t > M'$, and $\alpha' = \alpha t$. $S(t, M, \alpha) = \max\{OT(t), RT(t)\} - g(M, \alpha)$ denotes the interval time from $g(M, \alpha)$ to the earliest possible firing time of t. Let $\underline{S}(M, \alpha)$ be the lower bound of $S(t, M, \alpha)$ of all transitions enabled at (M, α). Then, the first selection function is defined as follows:

$$L(t, M, \alpha) = S(t, M, \alpha) + \underline{S}(M', \alpha').$$

The enabled transitions at (M, α) are sorted in an ascending order of magnitude $L(t, M, \alpha)$. If the number of enabled transitions is greater than max _*vertexes*, then only the top max _*vertexes* transitions with smaller $L(t, M, \alpha)$ values are considered to be fired.

Let $A(M, \alpha)$ denote the average value of $S(t, M, \alpha)$ on all transitions enabled at (M, α). Then the second selection function is defined as follows:

$$U(M, \alpha) = g(M, \alpha) + A(M, \alpha).$$

D. Deadlock-Free DWS Algorithm

A deadlock-free DWS algorithm (D^2WS) is now developed to find the optimal or suboptimal deadlock-free schedule. It is a search algorithm within the ERG of a PNS through dynamic windows. By using the proposed selection functions, the most promising vertexes are selected for further exploration. The movement of a search window is guided by Rules 1 and 2. If the final marking is reached, the optimal or suboptimal schedule is obtained and the corresponding sequence of transitions is output; otherwise, select an unexplored vertex and generate new vertexes from it. Rule 3 is used to decide whether a newly generated vertex should be further explored or not. Lists *OPEN* and *CLOSED* are used to store unexplored and explored vertexes, respectively. Now, we have algorithm D^{2}WS.

Algorithm D^{2}WS
Input: A PNS(N, M_0) and *high*, max _*size*, max _ *vertexes*, and max _*top*;
Output: Transition sequence σ;/* a feasible sequence of firing transitions */
Begin
Initialize: $OPEN = \{(M_0, \varepsilon)\}$; $CLOSED = \emptyset$;
$bottom\_depth = 0$; $top\_depth = high$;
while $(OPEN \neq \emptyset)$ do {
if $(\chi_u(bottom\_depth) = 0)$ {
$top\_depth = top\_depth + 1$;
$bottom\_depth = bottom\_depth + 1$;}/* Rule 2 */
select a (M, α) from *OPEN* with minimum $f(M, \alpha)$ and $depth(M, \alpha) < top\_depth$;
$OPEN := OPEN \backslash \{(M, \alpha)\}$;
$CLOSED := CLOSED \cup \{(M, \alpha)\}$;
let $ED(M) = \{t \in T | M[t >$ and $DS(M, t) =$ permitted }/*apply DAP to select safe transitions at M*/
sort $ED(M)$ in ascending order of magnitude $L(t, M, \alpha)$;
$Y(M, \alpha) = 0$; /* $Y(M, \alpha)$ records the number of successor vertexes of (M, α) */
while $(ED(M) \neq \emptyset)$ do{
let $t \in ED(M)$ be the first element in $ED(M)$;
$ED(M) := ED(M) \backslash \{t\}$;
let $M[t > M_1$, and $\alpha_1 = \alpha t$;
if $(M_1 = M_f)$\{ /* final mark M_f is reached */
Output: α_1; Exit;}
if (there is a node (M_1, α_2) in *OPEN* satisfying $U(M_1, \alpha_1) < U(M_1, \alpha_2)$){
$OPEN := (OPEN \backslash \{(M_1, \alpha_2)\}) \cup \{(M_1, \alpha_1)\}$;

else if (there is not a node with marking M_1 in *OPEN* $\cup$ *CLOSED* or there is a node (M_1, α_2) in *CLOSED* satisfying $U(M_1, \alpha_1) < U(M_1, \alpha_2)$ {
if (($\chi_e(depth(M_1, \alpha_1)) <$ max *_size*) or ($f(M_1, \alpha_1) <$ *BEST*$(depth(M_1, \alpha_1))$)) {
OPEN: = *OPEN* $\cup \{(M_1, \alpha_1)\}$;/* Rule 3*/
$Y(M, \alpha)$: = $Y(M, \alpha) + 1$;}
if (χ_u(*top_depth*) $\geq$ max *_top*){/* Rule 1*/
discard all vertexes at level *bottom_depth*;
top_depth = *top_depth* + 1;
bottom_depth = *bottom_depth* + 1;}
if ($Y(M, \alpha) \geq$ max *_vertexes*), break;
}/*end if-else if*/
}/*end while ($ED(M) \neq \emptyset$) * /
}/*end while (*OPEN* $\neq \emptyset$) * /
End

Theorem 4: D^{2}WS can always yield a solution.

The proof is given in the Appendix.

Note that the DS algorithm in [29] is embedded in D^{2}WS. It can prevent the system from safe markings to deadlock ones via a one-step look-ahead method. For a safe marking M and an enabled transition t at M, if firing t leads to a safe marking, then DS(M, t) = permitted and DS lets it fire; otherwise DS(M, t) = unpermitted and DS prohibits its firing.

Now let us illustrate the differences of the three heuristic functions via the following example.

Example 5: Consider the PNS in Fig. 1. Let $M_0 = p_{1s} + p_{2s} + 2r_1 + r_2$ be its initial marking and the processing time of parts in places p_{11}, p_{12}, p_{21}, and p_{22} be 1, 1, 2, and 2, respectively. Since here we only show few initial steps of the algorithm, we can suppose that the parameters of the window are big enough, i.e., all safe enabled transitions can be fired and all generated vertexes are included in the window.

Consider $f_k(M, \alpha) = g(M, \alpha) + h_k(M, \alpha)$, $k = 1$, 2, and 3. At the beginning, vertex (M_0, ε) is in *OPEN*. $ED(M_0) = \{t_{11}, t_{21}\}$. Let $M_0[t_{11} > M_1$ and $M_0[t_{21} > M_2$. For (M_1, t_{11}), $g(M_1, t_{11}) = 0$, $h_1(M_1, t_{11}) = h_2(M_1, t_{11}) = 2$, and $h_3(M_1, t_{11}) = 2.7$, and for (M_2, t_{21}), $g(M_2, t_{21}) = 0$ and $h_1(M_2, t_{21}) = h_2(M_2, t_{21}) = h_3(M_2, t_{21}) = 2$. By exploring (M_0, ε), we have two new vertexes (M_1, t_{11}) and (M_2, t_{21}) through firing t_{11} and t_{21}, respectively, and they are all added into *OPEN* yielding *OPEN* $= \{(M_1, t_{11}), (M_2, t_{21})\}$.

In *OPEN*, since $f_k(M_1, t_{11}) = f_k(M_2, t_{21}) = 2$, $k = 1$ and 2, either (M_1, t_{11}) or (M_1, t_{11}) can be first selected and explored if f_1 or f_2 is used in the algorithm. While $f_3(M_2, t_{21}) = 2 < f_3(M_1, t_{11}) = 2.7$. Thus, in *OPEN*, (M_2, t_{21}) must be first selected and explored if f_3 is used in D^{2}WS.

Thus, the search process under f_3 is different from those under f_1 and f_2. Now we illustrate only the difference between h_1 and h_2 as follows.

For *OPEN* $= \{(M_1, t_{11}), (M_2, t_{21})\}$, (M_1, t_{11}) is first explored. $ED(M_1) = \{t_{12}, t_{21}\}$. Firing t_{12} and t_{21} at (M_1, t_{11}) generate new vertexes $(M_3, t_{11}t_{12})$ and $(M_4, t_{11}t_{21})$, respectively. $f_1(M_3, t_{11}t_{12}) = f_2(M_3, t_{11}t_{12}) = 1 + 1.7 = 2.7$, $f_1(M_4, t_{11}t_{21}) = 0 + 2 = 2$, and $f_2(M_4, t_{11}t_{21}) = 0 + 2.7 = 2.7$. By adding $(M_3, t_{11}t_{12})$ and $(M_4, t_{11}t_{21})$ into *OPEN*, we have *OPEN* $= \{(M_2, t_{21}), (M_3, t_{11}t_{12}), (M_4, t_{11}t_{21})\}$. Then

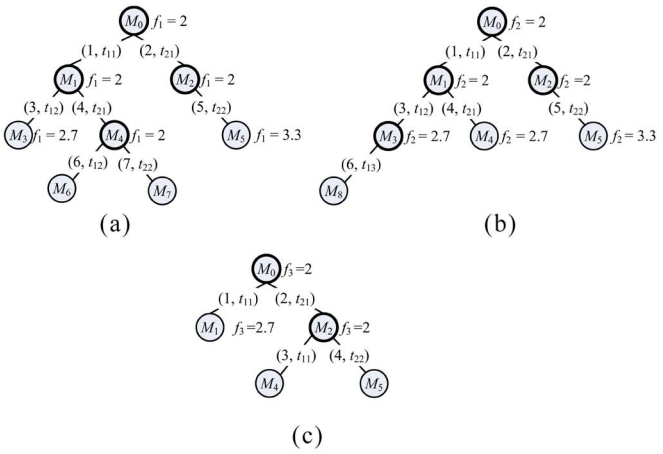

Fig. 3. Partial search graphs for PNS in Fig. 1 under different heuristic functions. (a) Under f_1. (b) Under f_2. (c) Under f_3.

(M_2, t_{21}) is the next vertex to be explored since $f_1(M_2, t_{21}) = f_1(M_4, t_{11}t_{21}) = 2 < f_1(M_3, t_{11}t_{12}) = 2.7$ and $f_2(M_2, t_{21}) = 2 < f_2(M_3, t_{11}t_{12}) = f_2(M_4, t_{11}t_{21}) = 2.7$.

At (M_2, t_{21}), $ED(M_2) = \{t_{11}, t_{22}\}$. Firing t_{11} and t_{22} at (M_2, t_{21}) generates $(M_4, t_{21}t_{11})$ and $(M_5, t_{21}t_{22})$, respectively. Because $(M_4, t_{11}t_{21})$ is already in *OPEN* and $U(M_4, t_{21}t_{11}) = U(M_4, t_{11}t_{21}) = 2$, $(M_4, t_{21}t_{11})$ is not added in *OPEN*. $f_1(M_5, t_{21}t_{22}) = f_2(M_5, t_{21}t_{22}) = 2 + 1.3 = 3.3$. So $(M_5, t_{21}t_{22})$ is added into *OPEN*, and then we have *OPEN* $= \{(M_3, t_{11}t_{12}), (M_4, t_{11}t_{21}), (M_5, t_{21}t_{22})\}$. Since $f_1(M_4, t_{11}t_{21}) = 2 < f_1(M_3, t_{11}t_{12}) = 2.7 < f_1(M_5, t_{21}t_{22}) = 3.3$, and $f_2(M_3, t_{11}t_{12}) = f_2(M_4, t_{11}t_{21}) = 2.7 < f_2(M_5, t_{21}t_{22}) = 3.3$, $(M_4, t_{11}t_{21})$ under f_1 and $(M_3, t_{11}t_{12})$ under f_2 are first selected to be explored.

In short, if f_1 is used in the algorithm, $t_{11}t_{21}t_{12}t_{21}t_{22}t_{12}t_{22}$ is first fired, while if f_2 is used, $t_{11}t_{21}t_{12}t_{21}t_{22}t_{13}t_{21}$ is first fired. Partial search graphs under three heuristic functions are shown in Fig. 3(a)–(c), where markings are in nodes, and transitions and their firing order are in arcs. For example, (k, t) denotes t is the k-th one fired. In Fig. 3, the nodes in bold are explored.

V. Experimental Results

The algorithm is implemented in C++ and run on a 2.19 GHz Microsoft Windows Sever with 3.99G RAM. In this section, D^{2}WS is tested first on an AMS studied in [27]. Its PNS model is shown in Fig. 4. Now, we have 8, 12, and 8 parts of q_1–q_3 types, respectively, to be processed, and the processing time is the same as that in [27], as shown in Table I.

The parameters of D^{2}WS are taken randomly distributed in the ranges of [2, 9] for *high*, [2, 5] for max *_size*, [2, 5] for max *_vertexes*, and [6, 15] for max *_top*. For simplicity, we take max *_top* $= 3 *$ max *_size*. Suppose that the capacities of m_1–m_4, and r_1–r_3 are 2, 2, 2, 2, 1, $C(r_2)$ and 1, respectively. According to $C(r_2) = 1$ or not, we will consider two cases. Under a combination of parameters, each case is solved three times using heuristic functions h_1–h_3, respectively.

Case 1: $C(r_2) = 2$. There is no ξ-resource in the system and hence DS can be used directly. We take ten random

TABLE I
PROCESSING TIME OF PART OPERATION OF THE AMS IN FIG. 4

q_1	q_2		q_3
w_1	w_2	w_3	w_4
o_{11}: 8	o_{21}: 4	o_{21}: 4	o_{31}: 5
o_{12}: 34	o_{22}: 32	o_{22}: 23	o_{32}: 22
o_{13}: 5	o_{23}: 8	o_{23}: 6	o_{33}: 4
	o_{24}: 38	o_{24}: 20	o_{34}: 17
	o_{25}: 5	o_{25}: 5	o_{35}: 6

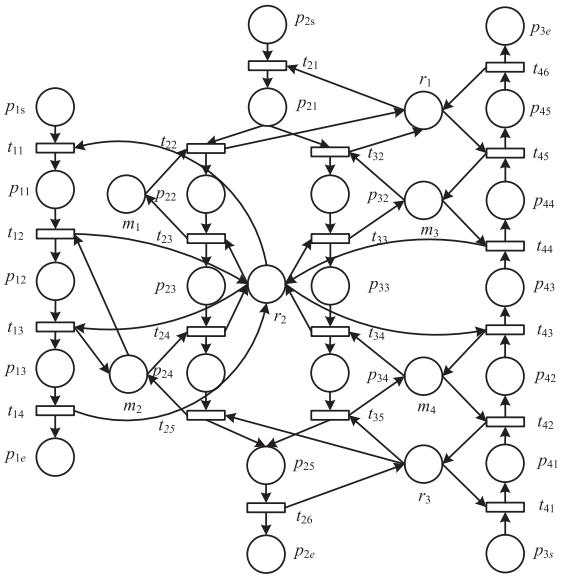

Fig. 4. PNS model of an AMS.

TABLE II
SIMULATION RESULTS FOR $C(r_2) = 2$

($high$, max_size, $max\_vertexes$, max_top)	Makespan			No.Explored vertexes		
	h_1	h_2	h_3	h_1	h_2	h_3
(2, 3, 3, 9)	308	295	295	733	738	748
(2, 3, 4, 9)	313	288	273	763	832	886
(3, 2, 2, 6)	276	268	272	364	362	378
(3, 3, 4, 9)	291	263	295	763	757	758
(4, 5, 2, 15)	292	292	270	937	952	956
(5, 3, 4, 9)	310	285	294	744	769	794
(6, 4, 3, 12)	316	312	300	939	885	943
(7, 4, 2, 12)	290	284	267	766	756	754
(8, 4, 2, 12)	323	284	284	767	770	784
(9, 4, 2, 12)	319	296	**255**	769	751	739

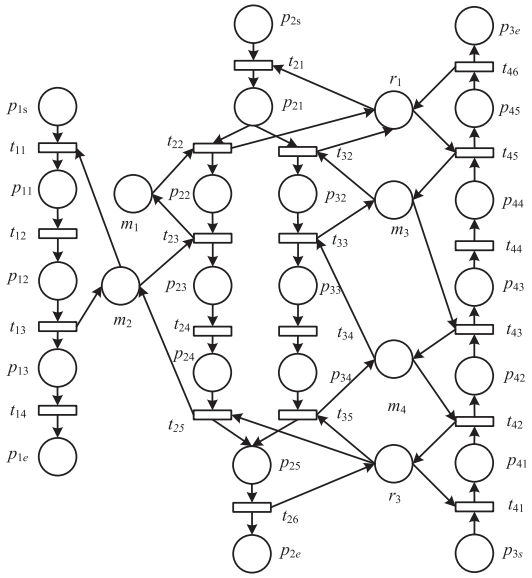

Fig. 5. Reduced PNS of PNS in Fig. 4.

TABLE III
SIMULATION RESULTS FOR $C(r_2) = 1$

($high$, max_size, $max\_vertexes$, max_top)	Makespan			No.Explored vertexes		
	h_1	h_2	h_3	h_1	h_2	h_3
(2, 3, 3, 9)	378	352	345	696	710	749
(2, 3, 4, 9)	384	353	307	741	752	767
(3, 2, 2, 6)	348	348	**303**	385	380	360
(3, 3, 4, 9)	379	353	332	704	734	715
(4, 5, 2, 15)	344	358	338	933	930	959
(5, 3, 4, 9)	377	327	324	703	752	728
(6, 4, 3, 12)	353	381	359	888	919	973
(7, 4, 2, 12)	362	353	343	753	781	759
(8, 4, 2, 12)	359	350	337	770	764	751
(9, 4, 2, 12)	353	374	354	762	770	792

TABLE IV
SUMMARY OF THE RESULTS

Results	Case1			Case2		
	h_1	h_2	h_3	h_1	h_2	h_3
Average makespan	303.8	286.7	280.5	363.7	354.9	334.2
Best makespan	276	263	255	344	327	303
Average explored vertexes _	754.5	758.2	773.5	733.5	749.2	755.3

combinations of parameters for testing, and the first solutions found by D^{2}WS under each combination are recorded and shown in Table II. The best deadlock-free schedule is obtained with makespan 255 under ($high$, max$_size$, max$_vertexes$, max$_top$) $= (9, 4, 2, 12)$ and the heuristic function h_3, and its corresponding sequence of transitions is t_{21}, t_{41}, t_{22}, t_{21}, t_{42}, t_{11}, t_{11}, t_{32}, t_{41}, t_{12}, t_{11}, t_{12}, t_{21}, t_{42}, t_{22}, t_{41}, t_{21}, t_{43}, t_{44}, t_{33}, t_{32}, t_{34}, t_{43}, t_{42}, t_{41}, t_{45}, t_{44}, t_{13}, t_{12}, t_{11}, t_{14}, t_{13}, t_{46}, t_{21}, t_{12}, t_{33}, t_{32}, t_{21}, t_{14}, t_{43}, t_{42}, t_{35}, t_{34}, t_{23}, t_{22}, t_{45}, t_{44}, t_{26}, t_{41}, t_{43}, t_{42}, t_{46}, t_{35}, t_{45}, t_{44}, t_{13}, t_{24}, t_{11}, t_{26}, t_{41}, t_{46}, t_{21}, t_{42}, t_{14}, t_{13}, t_{41}, t_{12}, t_{33}, t_{32}, t_{14}, t_{21}, t_{43}, t_{34}, t_{23}, t_{22}, t_{45}, t_{46}, t_{21}, t_{44}, t_{43}, t_{42}, t_{25}, t_{24}, t_{33}, t_{32}, t_{45}, t_{44}, t_{11}, t_{26}, t_{35}, t_{34}, t_{13}, t_{12}, t_{14}, t_{11}, t_{11}, t_{26}, t_{46}, t_{45}, t_{46}, t_{35}, t_{21}, t_{32}, t_{21}, t_{26}, t_{41}, t_{42}, t_{25}, t_{12}, t_{26}, t_{13}, t_{12}, t_{33}, t_{32}, t_{14}, t_{43}, t_{34}, t_{33}, t_{44}, t_{21}, t_{43}, t_{34}, t_{33}, t_{32}, t_{45}, t_{44}, t_{23}, t_{35}, t_{34}, t_{13}, t_{14}, t_{13}, t_{24}, t_{23}, t_{24}, t_{46}, t_{45}, t_{26}, t_{33}, t_{35}, t_{14}, t_{46}, t_{34}, t_{26}, t_{35}, t_{26}, t_{35}, t_{26}, t_{25}, t_{26}, t_{25}, t_{26}.

Case 2: $C(r_2) = 1$. r_2 is a ξ-resource. Its reduced model $(N(r_2), M_{A0})$, which is shown in Fig. 5, has no ξ-resource.

Thus, for $(N(r_2), M_{A0})$, the optimal DAP ρ^* can be obtained by DS. The system of $(N(r_2), M_{A0})$ controlled by ρ^*, $\rho^*||(N(r_2), M_{A0})$, is a deadlock avoidance supervisor of (N, M_0).

We take the same parameter combinations as in Case 1 for testing, and the results are shown in Table III. The best deadlock-free schedule is obtained with makespan 303 under ($high$, max$_size$, max$_vertexes$, max$_top$) $= (3, 2, 2, 6)$ and the heuristic function h_3.

The summary of the results of the two cases is shown in Table IV. In most cases, the run time under different parameter

TABLE V
SCHEDULING RESULTS VIA D^{2}WS AND GA2

	D^{2}WS		E+GA2		L+GA2		H+GA2		X+GA2		P+GA2	
Instance	BST	AVG	BST	AVG	BST	AVG	BST	AVG	BST	AVG	BST	AVG
*In*01	**303**	**334.2**	420	448.9	416	449.2	338	372.4	352	400.7	331	364.1
*In*02	**426**	**454.9**	645	690.8	644	688.2	525	568.4	531	588.5	475	524.5
*In*03	**539**	**581.1**	760	833.4	784	836.4	648	723.0	717	764.2	612	691.1
*In*04	**672**	**739.7**	907	965.6	895	957.5	837	864.1	890	925.3	803	858.6
*In*05	**252**	**272.6**	294	316.6	319	356.4	\	\	254	275.7	\	\
*In*06	**360**	**388.6**	438	480.0	510	536.5	\	\	370	404.7	\	\
*In*07	**438**	**483.1**	545	569.9	601	645.4	\	\	485	518.3	\	\
*In*08	**569**	**593.5**	629	686.9	691	768.4	\	\	602	648.2	\	\
*In*09	194	207.6	219	238.4	239	251.8	\	\	**184**	**202.0**	\	\
*In*10	**269**	**286.5**	327	352.7	356	388.4	\	\	278	300.5	\	\
*In*11	**343**	**367.1**	404	435.7	437	467.5	\	\	356	378.4	\	\
*In*12	**403**	**440.2**	490	515.7	523	577.2	\	\	426	454.5	\	\
*In*13	182	193	191	203.1	209	222.3	\	\	**170**	**182.1**	\	\
*In*14	**246**	**261.5**	285	308.5	318	338.9	\	\	247	263.2	\	\
*In*15	**305**	**330.6**	347	371.5	406	429.6	\	\	332	349.6	\	\
*In*16	**359**	**385.5**	418	440.7	475	525.0	\	\	389	425.1	\	\

combinations is less than 600 s. The shortest one is 231 s with the parameter combination $(3, 2, 2, 6)$ and h_3 for testing Case 2, and the longest one is 958 s with the combination $(6, 4, 3, 12)$ and h_3 for Case 2.

For two cases, the best solutions obtained in [27], have makespan 321 and 447, respectively. By comparing with the best makespan in [27], our algorithm achieves much better results. For Case 1, it is interesting that average makespan for h_1–h_3 is much smaller than the best makespan 321 in [27]. For Case 2, the largest makespan for h_1–h_3 is 384, which is smaller than the best makespan 447 in [27]. The results indicate the effectiveness of the proposed D^{2}WS.

From simulation results in Tables II and III, we find that D^{2}WS under most parameter combinations can obtain better solutions by using h_2 than h_1. From the summary of results in Table IV, we know that both the best and average makespan obtained by using h_2 for Cases 1 and 2 are smaller than those by using h_1. The simulation results show that the scheduling performance under different admissible heuristic functions is consistent with their informedness.

From Table IV, it is easy to find that both the best and average makespan using h_1–h_3 for Cases 1 and 2 are in a descending order. This implies the importance of considering the idle time in designing a heuristic function. Moreover, a reasonable inadmissible heuristic function may outperform admissible ones in a partial space search algorithm, e.g., D^{2}WS.

Now, let us compare D^{2}WS with the genetic algorithm (GA2) presented in [8]. In [8], 16 instances based on the PN model as shown in Fig. 4 are listed. They can be divided into four groups according to the resource capacities ($C(m_i), i = 1, 2, 3, 4$, and $C(r_j), j = 1, 2, 3$), while each group has four instances with the numbers of type-q_1, q_2, and q_3 $(8, 12, 8)$, $(10, 20, 10)$, $(15, 20, 15)$, and $(20, 20, 20)$, respectively.

1) *In*01-*In*04: $C(m_i) = 2$ and $C(r_i) = 1$.
2) *In*05-*In*08: $C(m_i) = 2$ and $C(r_i) = 2$.
3) *In*09-*In*12: $C(m_i) = 3$ and $C(r_i) = 2$.
4) *In*13-*In*16: $C(m_i) = 3$ and $C(r_i) = 3$.

Scheduling results obtained by using D^{2}WS and GA2 [8] are listed in Table V, where h_3 is applied in D^{2}WS, and each instance is tested ten times with ten random combinations of parameters which have the same range as described as above, respectively; while five different deadlock controllers are embedded into GA2, resulting in five different algorithms, namely E+GA2, L+GA2, H+GA2, X+GA2, and P+GA2, and their computing results are from [8].

Most of the run time for different parameters and instances is less than 800 s. The shortest one is 178 s with the parameter combination $(5, 2, 2, 6)$ for testing *In*13, and the longest one is 1900 s with the combination $(4, 5, 3, 15)$ for *In*16.

For 14 instances out of 16, excellent solutions (in bold) are found by D^{2}WS and are better than the ones found by GA2 with different controllers in [8]. All of the best and average solutions found by D^{2}WS are better than those by E+GA2 and L+GA2 [8].

For the instances with larger lot sizes [larger than $(8, 12, 8)$], D^{2}WS outperforms GA2 with different controllers. That shows the excellent ability of D^{2}WS in finding optimal or near-optimal solutions for larger scale problems. The excellent performance of D^{2}WS can be attributed to the quality of the proposed heuristic functions.

VI. CONCLUSION

A deadlock-free hybrid search algorithm (D^{2}WS) is proposed for the scheduling problem of AMSs with shared resources and route flexibility. Based on the extended RG of a place-timed PN, it finds optimal or near-optimal deadlock-free schedules at given initial markings. Deadlocks are avoided in the scheduling during the construction of the graph by selecting enabled transitions under the guidance of polynomial-complexity DAPs. To increase the chance of finding the optimal solution, three heuristic and two selection functions are newly established to guide a search process. D^{2}WS overcomes the state space explosion problem of classical heuristic searches based on state space exploration by restraining the current search within a local search window. Since it has adopted some strategies to restrict the search space and DAPs are polynomially complex, it is also suitable for large-scale systems.

The performances of three different heuristic functions in D^{2}WS are evaluated through a realistic AMS in [27]. Average makespan for different heuristic functions is much smaller than the best one in [27]. D^{2}WS takes less than 600 s for most cases. The third heuristic function is the most powerful. Then, D^{2}WS with the most powerful heuristic function is used to test on 16 AMS instances. Computational results demonstrate that the solutions of high quality can be found in an acceptable time. For 14 instances out of 16, solutions found by D^{2}WS are better than those found by genetic algorithms with different controllers in [8]. D^{2}WS with different parameters for most instances takes less than 800 s.

The simulation results show that a heuristic function is very important to intelligent search algorithms such as D^{2}WS, and the performance of D^{2}WS under different admissible heuristic functions is consistent with their informedness. Moreover, a reasonable inadmissible heuristic function may outperform admissible ones in a partial space search algorithm. The excellent performance of D^{2}WS can be attributed to the quality of our newly proposed heuristic functions. The simulation results also show that, for larger-scale problems, D^{2}WS has desired performance in finding optimal or near-optimal schedules. Its application to the scheduling problems of failure and deadlock-prone manufacturing systems will be investigated in our future work.

APPENDIX

PROOF OF THEOREMS 2 AND 4

Proof of Theorem 2: $\sum_{r\in ER} d_1(M,\alpha,\sigma*,r)$ is the sum of all idle time of all resources from $g(M,\alpha)$ to the end, while $\sum_{r\in R}\delta(r,M,\alpha)G(r,M,\alpha)$ is only part of the sum of idle time of all resources. Thus, $\sum_{r\in R}\delta(r,M,\alpha)G(r,M,\alpha) \le \sum_{r\in ER} d_1(M,\alpha,\sigma*,r))$.

Since $\sum_{j(p,i)\in J}(RT(j(p,i),M,\alpha) + X(p))$ is the sum of the minimum total processing time of all parts from $g(M,\alpha)$ to the end, $\sum_{j(p,i)\in J}(RT(j(p,i),M,\alpha) + X(p)) \le \sum_{r\in ER} d_2(M,\alpha,\sigma*,r)$. Then $h_2(M,\alpha) = \sum_{r\in R}\delta(r,M,\alpha)G(r,M,\alpha) + \sum_{j(p,i)\in J}(RT(j(p,i),M,\alpha) + X(p)) \le \sum_{r\in ER} d_1(M,\alpha,\sigma*,r)) + \sum_{r\in ER} d_2(M,\alpha,\sigma*,r) = h*(M,\alpha)$, and hence, h_2 is admissible.

Since $\forall r \in R$, $G(r,M,\alpha) \ge 0$, $\sum_{r\in R}\delta(r,M,\alpha)$ $G(r,M,\alpha)/|ER| \ge 0$, and hence, $h_1(M,\alpha) \le h_2(M,\alpha)$. Then $h_1(M,\alpha) \le h*(M,\alpha)$ and hence, h_1 is admissible. ■

Proof of Theorem 4: DAPs [29] prohibit transition firings that lead the PNS from safe markings to deadlock and all vertexes in D^{2}WS are generated under DAPs. The markings of all generated vertexes are safe. In other words, from any one of them unless it is M_f, there is at least an enabled transition that can fire and lead to a safe marking.

There are two termination conditions for D^{2}WS: the final marking is reached or $OPEN = \emptyset$. If the program always terminates with the final marking reached, then the program is correct. Suppose that the final marking has not been reached but the program terminates with $OPEN = \emptyset$. Then, all vertexes in the current window (upon termination of the algorithm) are explored. Let (M,α) be an explored vertex with the deepest depth among all the vertexes in the current window. Then, since $M \neq M_f$ and M is safe, there must be a vertex (M_1,α_1) in the current window generated from (M,α) and $depth(M_1,\alpha_1) = depth(M,\alpha)+1$. (M_1,α_1) will be kept in the current window unless there is a vertex at $depth(M_1,\alpha_1)$ with marking M_1 or the number of successor vertexes of (M,α) is more than $\max\_size$. That is a contradiction to the assumption that (M,α) is an explored vertex with the deepest depth among all the vertexes in the current window. Thus, D^{2}WS can always end with the final marking. ■

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.

REFERENCES

[1] B. Abdallah, H. A. Elmaraghy, and T. Elmekkawy, "Deadlock-free scheduling in flexible manufacturing systems using Petri nets," *Int. J. Prod. Res.*, vol. 40, pp. 2733–2756, Feb. 2002.

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

[3] Y. Dashora, S. Kumar, M. K. Tiwari, and S. T. Newman, "Deadlock-free scheduling of an automated manufacturing system using an enhanced colored time resource Petri-net model-based evolutionary endosymbiotic learning automata approach," *Int J. Flexible Manuf. Syst.*, vol. 19, pp. 486–515, Dec. 2007.

[4] 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.

[5] G. Xu 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.

[6] H. R. Golmakani, J. K. Mills, and B. Benhabib, "Deadlock-free scheduling and control of flexible manufacturing cells using automata theory," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 2, pp. 327–337, Mar. 2006.

[7] O. Y. Gusikhin and A. S. Kulinich, "Animated AI-based simulation in production scheduling," in *Proc. IFIP TC5/WG5. 3/IFAC. Int. Workshop Knowl. Hybrid Syst. Eng. Manuf.*, 1993, pp. 165–176.

[8] 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, pp. 1557–1572, Oct. 2014.

[9] 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, pp. 283–305, Jan. 2001.

[10] B. Huang, X. X. Shi, and N. Xu, "Scheduling FMS with alternative routings using Petri nets and near admissible heuristic search," *Int. J. Adv. Manuf. Technol.*, vol. 63, pp. 1131–1136, Feb. 2012.

[11] B. Huang, Y. Sun, and Y. M. Sun, "Scheduling of flexible manufacturing systems based on Petri nets and hybrid heuristic search," *Int. J. Prod. Res.*, vol. 46, pp. 4553–4565, Aug. 2008.

[12] B. Huang, Y. Sun, Y. M. Sun, and C. X. Zhao, "A hybrid heuristic search algorithm for scheduling FMS based on Petri net model," *Int. J. Adv. Manuf. Technol.*, vol. 48, nos. 9–12, pp. 925–933, 2010.

[13] M. D. Jeng and S. C. Chen, "A heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems," *Int. J. Flexible Manuf. Syst.*, vol. 10, no. 2, pp. 139–162, 1998.

[14] D. Y. Lee and F. DiCesare, "Scheduling flexible manufacturing systems using Petri nets and heuristic search," *IEEE Trans. Robot. Autom.*, vol. 10, no. 2, pp. 123–132, Apr. 1994.

[15] J. Lee and J. S. Lee, "Heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix," *Comput. Ind. Eng.*, vol. 59, no. 2, pp. 799–806, Aug. 2010.

[16] 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.
[17] A. R. Moro, H. Yu, and G. Kelleher, "Advanced scheduling methodologies for flexible manufacturing systems using Petri nets and heuristic search," in *Proc. IEEE Int. Robot. Autom.*, San Francisco, CA, USA, 2000, pp. 2398–2403.
[18] L. Pan, Z. Ding and M. C. Zhou, "A configurable state class method for temporal analysis of time Petri nets," *IEEE Trans. Syst., Man, and Cybern., Syst.*, vol. 44, no. 4, pp. 482–493, Apr. 2014.
[19] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA, USA: Addison-Wesley, 1984.
[20] 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.
[21] 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.
[22] A. Reyes, H. Yu, G. Kelleher, and S. Lloyd, "Integrating Petri nets and hybrid heuristic search for the scheduling of FMS," *Comput. Ind.*, vol. 47, no. 1, pp. 123–138, 2002.
[23] T. Sun, C. Cheng, and L. Fu, "A Petri net based approach to modeling and scheduling for an FMS and a case study," *IEEE Trans. Robot. Autom. Mag.*, vol. 41, no. 6, pp. 593–601, Dec. 1994.
[24] S. Tzafestas and A. Triantafyllakis, "Deterministic scheduling in computing and manufacturing systems: A survey of models and algorithms," *Math. Comput. Simul.*, vol. 35, no. 5, pp. 397–434, Nov. 1993.
[25] 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.*, vol. 6, no. 6, pp. 713–723, Dec. 1990.
[26] 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, 2007.
[27] 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.
[28] 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.
[29] 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.
[30] H. H. Xiong and M. C. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 384–393, Aug. 1998.
[31] H. H. Xiong, M. C. Zhou, and R. J. Caudill, "A hybrid heuristic search algorithm for scheduling flexible manufacturing systems," in *Proc. IEEE Int. Robot. Autom.*, Albuquerque, NM, USA, 1996, pp. 2793–2797.
[32] S. J. Yim and D. Y. Lee, "Multiple objective scheduling for flexible manufacturing systems using Petri nets and heuristic search," in *Proc. IEEE Int. Syst., Man, Cybern.*, Beijing, China, 1996, pp. 2984–2989.
[33] H. J. Yoon and D. Y. Lee, "Deadlock-free scheduling of photolithography equipment in semiconductor fabrication," *IEEE Trans. Semicond. Manuf.*, vol. 17, no. 1, pp. 42–54, Feb. 2004.
[34] H. Yu, A. Reyes, S. Cang, and S. Lloyd, "Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems—Part I. Petri net modelling and heuristic search," *Comput. Ind. Eng.*, vol. 44, pp. 527–543, Apr. 2003.
[35] H. Yu, A. Reyes, S. Cang, and S. Lloyd, "Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems—Part II. Heuristic hybrid search," *Comput. Ind. Eng.*, vol. 44, pp. 545–566, Apr. 2003.
[36] M. C. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Singapore: World Scientific, 1998.
[37] 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.
[38] 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.
[39] N. 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.
[40] N. 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, Sept. 2013.

JianChao Luo received the B.S. degree in electrical engineering and its automation from Chang'an University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree from the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an.

His current research interests include scheduling and control of automated manufacturing systems.

KeYi Xing (M'07) received the B.S. degree in mathematics from Northwest University, in 1982, the M.S. degree in applied mathematics from Xidian University, in 1985, and the Ph.D. degree in systems engineering from Xi'an Jiaotong University, in 1994 all in Xi'an, China.

He was with Xidian University in 1985. Since 2004, he has been with Xi'an Jiaotong University, where he is currently a Professor of Systems Engineering with the State Key Laboratory for Manufacturing Systems Engineering and the Systems Engineering Institute. His current research interests include control and scheduling of automated manufacturing, discrete event, and hybrid systems.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and 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, where he is currently a Distinguished Professor of Electrical and Computer Engineering. His current research interests include Petri nets, sensor networks, semiconductor manufacturing, transportation, and energy systems. He has authored or co-authored over 550 publications, including 11 books, and over 260 journal papers (majority in IEEE Transactions), and 22 book-chapters.

Prof. Zhou is the Founding Editor of the IEEE Press Book Series on Systems Science and Engineering. He is a Life Member of the Chinese Association for Science and Technology-USA and served as its President in 1999. He is a fellow of the IFAC and AAAS.

XiaoLing Li received the B.S. degree in electronic science and technology from Xi'an Jiaotong University, Xi'an, China, in 2012, where she is currently pursuing the Ph.D. degree from the Systems Engineering Institute.

Her current research interests include control and scheduling of automated manufacturing systems and discrete-event systems.

XinNian Wang received the B.S. degree in automation science and technology from Xi'an Jiaotong University, Xi'an, China, in 2011, where he is currently pursuing the Ph.D. degree from the Systems Engineering Institute.

His current research interests include control and scheduling of automated manufacturing systems.