Scheduling AMSs with Generalized Petri Nets and Highly Informed Heuristic Search

FengLian Yuan [a], Bo Huang [a,*], JianYong Lv [a], XiaoYu Lu [b]

[a] *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*

[b] *School of Cyber Science and Engineering, Nanjing University of Science and Technology, Jiangyin 214443, China*

Abstract

The design of the heuristic function in Petri-net(PN)-based A* search has a significant impact on the efficiency and schedule quality for automated manufacturing systems (AMSs). In Luo *et al.* (2015), two admissible heuristic functions are formulated for an A* search rooted in place-timed PNs to schedule AMSs. To increase application scenarios and improve search efficiency, this paper proposes a new heuristic function whose calculations take account of multiple resource acquisitions, weighted arcs, redundant resource units, and outdated resources, which are commonly encountered in practical AMSs but usually not considered. In addition, it is proven to be admissible and more informed than its counterparts. This quality ensures that the obtained schedules are optimal, while making its timed PN-based A* search more efficient. To ascertain the efficacy and efficiency of the proposed method, several benchmark systems are tested.

Key words: Automated manufacturing system; heuristic function; heuristic search; scheduling; timed Petri net.

1 Introduction

Automated manufacturing systems (AMSs) frequently involve a multitude of options for resource allocation and system execution paths, enabling high production rates. This presents a complex problem in scheduling, as it necessitates the rational arrangement of various resources and operations to attain optimal system efficiency. The scheduling problem encountered in AMSs is known to be NP-hard (Huang & Zhou, 2020; Pinto *et al.*, 2024). Petri Nets (PNs) constitute a suitable mathematical framework for modeling and analyzing automated systems, encapsulating fundamental aspects including sequence, concurrency, conflict, and synchronization (Mejía *et al.*, 2016). Moreover, the reachability graphs (RGs) derived from PNs aptly depict system behavior (Uzam *et al.*, 2016; Mejía & Pereira, 2020). The pioneering work of Lee & DiCesare (1994) introduced the application of the intelligent A^* search for system scheduling, utilizing reachability graphs from place-timed PNs. This approach entails traversing a partial RG rather than the entire graph to obtain an optimal schedule, facilitated by an admissible heuristic function, which provides an estimate of the remaining cost from a marking to a given goal marking without exceeding the actual minimal cost. Such methodology has proven effective in solving AMS scheduling tasks.

However, the efficacy of an A^* search heavily relies on the quality of its used heuristic function, which significantly impacts both the quality of results and search efficiency (Huang *et al.*, 2023). A highly informed admissible heuristic function enhances the efficiency of an A^* search, thereby aiding in the pursuit of an optimal schedule. In Luo *et al.* (2015), a new deadlock-free scheduling method that combines deadlock control policies and an A* search strategy in the RG of a place-timed Petri net (PN) is proposed to schedule AMSs. Two admissible heuristic functions are formulated to guide the PN-based A* search. One uses the average minimum time for a resource unit to complete its remaining operations from the current state S to a given goal state S_G as S's heuristic value. The other is similar to the first one except that it includes the sum of the minimal idle time of all resources, which is essential for the state

⋆ This work was in part supported by the Jiangsu Province Special Funds for Technology Transfer Projects under Grant BA2023022.

* Corresponding author: Tel.: +86 25 84315660, Fax: +86 25 84315960.

Email addresses: yuangfenglian@njust.edu.cn (FengLian Yuan), huangbo@njust.edu.cn (Bo Huang), Lv_jy@njust.edu.cn (JianYong Lv), xiaoyu.lu@njust.edu.cn (XiaoYu Lu).

transfer from S to S_G. So, it is more informed than the first one, which leads to a faster A* search in the RG of a place-timed PN.

The heuristic function used by an A* search is critical to application scenarios and search efficiency. However, the heuristic functions in Luo *et al.* (2015) are designed for extended S^3PR nets of AMSs, which are ordinary nets whose arc weights cannot be greater than one, and do not account for multiple resource acquisitions, weighted arcs, redundant resource units, and outdated resources, which are commonly encountered in the PNs for practical AMSs and should be considered in the A* search process. This paper aims to address the limitations and rectify the above issues.

In this work, a novel heuristic function for the PN-based A* search is proposed. It takes account of the above all cases in its state heuristic evaluations, and it has the following merit:

- It can be applied not only to ordinary PNs but also to generalized ones for AMSs, including S^4PR, S^5PR, S^*PR, PC^2R, etc (Huang & Zhou, 2020).
- It can deal with multiple resource acquisitions, weighted arcs, redundant resource units, and outdated resources, which are common in the PNs of AMSs.
- It is proven admissible and more informed than its counterparts in Luo *et al.* (2015), which makes its obtained schedules optimal and its PN-based A* search more efficient.

The structure of the paper is as below. Section 2 gives an introduction to the fundamental definitions of PNs, place-timed PNs, and PN-based A* searches. Section 3 presents our new admissible heuristic function for the PN-based A* searches and its properties. In Section 4, several benchmark PNs are examined, and their results are analyzed. Section 5 discusses the possibilities of combining the method with different techniques to handle bigger problems. Finally, in Section 6, we present the concluding remarks of this paper.

2 Basic Concepts

This section offers a summary of the definitions and notations related to PNs and place-timed PNs used for scheduling AMSs. In addition, the place-timed PN-based A* search is reviewed. For further details on them, please refer to (Huang *et al.*, 2023).

2.1 PNs

Let $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$ and $\mathbb{Z} = \{0\} \cup \mathbb{Z}^+$. PNs are a mathematical modeling tool for discrete event systems. They are usually defined as a four-tuple $N = (P, T, F, W)$ in which P and T are respectively the finite sets of places and transitions, satisfying $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ includes directed arcs between P and T. The function $W : F \to \mathbb{Z}^+$ designates weights to all the arcs. A PN N is considered ordinary if $\forall (x, y) \in F$, $W(x, y) = 1$. N is classified as generalized if $\exists (x, y) \in F$, $W(x, y) > 1$.

For a transition $t \in T$, ${}^\bullet t = \{p \in P | (p, t) \in F\}$ is its set of pre-places, and $t^\bullet = \{p \in P | (t, p) \in F\}$ is its set of post-places. Likewise, $\forall p \in P$, ${}^\bullet p = \{t \in T | (t, p) \in F\}$ denotes its set of pre-transitions, and $p^\bullet = t \in T | (p, t) \in F$ denotes its set of post-transitions.

In the context of a PN N, $M : P \to \mathbb{Z}$ denotes a marking of N, and $M(p)$ signifies the count of tokens in $p \in P$ at M. M_0 denotes the net's initial marking. At a given marking M, for a transition t, if $\forall p \in {}^\bullet t$, $M(p) \geqslant W(p, t)$, then t is enabled at M. When an enabled transition t fires at M, it produces another marking M' satisfying that $\forall p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$. If sequentially firing several transitions from M finally generates another marking M', we say that M' is reachable from M. All the markings reachable from M are represented as $\mathcal{R}(N, M)$.

A pair (N, M_0) denotes a net system. For (N, M_0), $\mathcal{G}(N, M_0)$ is called its RG, which is a state evolution graph including all reachable markings and edges, each of which transfers the net from one marking to another via a transition firing. (N, M_0) is considered $k-$bounded if $\forall M \in \mathcal{R}(N, M_0)$, $\forall p \in P$, $\exists k \in \mathbb{Z}^+$, $M(p) \leq k$. For (N, M_0), a place invariant $I : P \to \mathbb{Z}$ is defined as a $|P|$-dimensional integer vector satisfying the conditions $I \neq \mathbf{0}$, $I^T[N] = \mathbf{0}^T$, and $\forall M \in \mathcal{R}(N, M_0)$, $I^T M = I^T M_0$, in which $[N] = [N]^+ - [N]^-$ is a $|P| \times |T|$ incidence matrix of N satisfying that $[N]^+(p, t) = W(t, p)$ and $[N]^-(p, t) = W(p, t)$, and $\mathbf{0}$ denotes a zero vector. I becomes a P-semiflow if all of its elements are non-negative.

2.2 Place-timed PNs for System Scheduling

Within the literature, various kinds of PNs have been developed to address the issue of deadlock control in AMSs. Notable examples include S^4PR, S^5PR, S^*PR, and PC^2R. These PN classes can be transformed into place-timed generalized PNs for system scheduling of AMSs, as shown by Huang (Huang & Zhou, 2020). The conversion process entails partitioning every idle place into a start place and an end place within each job, transferring tokens from idle places to their respective start places, and incorporating time delays into activity places.

Definition 1 *A generalized place-timed PN for scheduling purposes is defined as* $\mathcal{N} = (P_S \cup P_E \cup P_R \cup P_A, T, F, W, D)$, *where it is composed of some process subnets* $\mathcal{N}^x = (P_S^x \cup P_E^x \cup P_R^x \cup P_A^x, T^x, F^x, W^x, D^x)$

that share some places, i.e., $\mathcal{N} = \cup_{x\in\mathbb{N}_J}\mathcal{N}^x$ *in which* $\mathbb{N}_J = \{i \in \mathbb{Z}^+ | \mathcal{N}^i$ *is the ith subnet of* $\mathcal{N}\}$ *and the subsequent assertions are valid.*

(1) For a process subnet $\mathcal{N}^x$, $P^x = P_S^x \cup P_E^x \cup P_A^x \cup P_R^x$ *where* P_S^x, P_E^x, P_A^x, *and* P_R^x *are the sets of start places, end places, activity places, and resource places of* $\mathcal{N}^x$, *respectively. Both* P_S^x *and* P_E^x *contain one place.*

(2) $P_A^x \neq \emptyset$, $P_R^x \neq \emptyset$, $P_S^x \cap P_E^x = \emptyset$, $(P_S^x \cup P_E^x) \cap P_A^x = \emptyset$, *and* $(P_S^x \cup P_E^x \cup P_A^x) \cap P_R^x = \emptyset$.

(3) $W = W_A \cup W_R$ *with* $W_A : F \cap ((P_A \cup P_S \cup P_E) \times T) \cup (T \times (P_A \cup P_S \cup P_E)) \to \{1\}$ *and* $W_R : F \cap ((P_R \times T) \cup (T \times P_R)) \to \mathbb{Z}^+$.

(4) $\forall r \in P_R$, *there is a unique minimal P-semiflow* I_r *satisfying that* $\|I_r\| \cap P_R = \{r\}$, $\|I_r\| \cap (P_A \cup P_S \cup P_E) \neq \emptyset$, *and* $I_r(r) = 1$. *In addition,* $P_A \subseteq \bigcup_{r\in P_R}(\|I_r\| \setminus \{r\})$.

(5) The places shared by $\mathcal{N}^x$ *and* $\mathcal{N}^y (x \neq y)$ *constitute a subset of* P_R.

(6) $D : P_A \to \mathbb{Z}$ *represents the assignment of deterministic operation time to* P_A.

In Definition 1, Item 1 states that every process subnet possesses a start place and an end place. Item 3 specifies that the weight assigned to the arcs that connect transitions with activity places, start places, or end places is uniformly set to one. However, the weight of the arcs that connect transitions with resource places is allowed to exceed one, representing the scenarios where operations may require multiple resources. Item 4 imposes resource preservation constraints on specific places, including resource places themselves. Item 5 signifies the sharing of certain resource places among different process subnets. Lastly, Item 6 introduces deterministic time information associated with activity places. Note that the scheduling problem necessitates the utilization of PNs with time information. More specifically, such PNs are classified into timed PNs or time ones. Timed PNs are characterized by their use of time durations, and time PNs rely on time intervals for modeling. Timed PNs have two categories: deterministic timed PNs, which employ fixed time durations, and stochastic timed PNs, which involve time durations with inherent randomness (Huang & Zhou, 2020). This study focuses on the application of deterministic place-timed PNs.

2.3 PN-based A Search*

After incorporating time information into a place-timed PN $\mathcal{N}$, each token residing in a place $p \in P_A$ is required to remain in p for a minimal duration of $D(p)$ time units before it becomes eligible to engage in the firing of a p's post-transition. As a result, in such a net, the firing of an enabled transition not only alters M but also impacts the remaining time associated with each token located in activity places.

Definition 2 *For a place-timed net system* $(\mathcal{N}, S_0)$ *with k-bounded activity places, a state is defined as* $S = (M, R)$ *where* M *is a token distribution vector with* $|P|$ *elements and* R *is a* $|P| \times k$ *remaining token time matrix.*

Accordingly, for a place-timed net system $(\mathcal{N}, S_0)$, we utilize $\mathcal{R}(\mathcal{N}, S_0)$ to represent the set of its reachable states and use $\mathcal{G}(\mathcal{N}, S_0)$ to denote its RG. Let $c(S, S')$ be the time required for the transfer from S to S'. The place-timed PN-based A* algorithm (Huang *et al.*, 2023), which searches for a sequence of transition firings from S_0 to a goal state S_G in $\mathcal{G}(\mathcal{N}, S_0)$, is described as Algorithm 1.

Algorithm 1 PN-based A* Search.

Input: A place-timed PN $(\mathcal{N}, S_0)$ with S_0 and S_G.
Output: A transition firing sequence leading from S_0 to S_G.

Put S_0 in OPEN;
while True **do**
  if OPEN= $\emptyset$ **then**
    terminate with failure;
  end if
  move the first state $S = (M, R)$ of OPEN to CLOSED;
  if S equals S_G **then**
    produce a path from S back to S_0 and terminate;
  end if
  for each t enabled at M **do**
    decrease all remaining token time to enable t at both M and R;
    obtain a child state S' by firing t;
    set a pointer from S' to S;
    calculate $g(S') = g(S) + c(S, S')$, $h(S')$, and $f(S') = g(S') + h(S')$;
    if there exists S^O in OPEN such that $M(S') = M(S^O)$, $R(S') = R(S^O)$, and $g(S') < g(S^O)$ **then**
      replace S^O with S' in OPEN;
    end if
    if there exists S^C in CLOSED satisfying that $M(S') = M(S^C)$, $R(S') = R(S^C)$, and $g(S') < g(S^C)$ **then**
      delete S^C from CLOSED and insert S' into OPEN;
    end if
    if there exists S'' in neither OPEN nor CLOSED such that $M(S') = M(S'')$ and $R(S') = R(S'')$ **then**
      insert S' into OPEN;
    end if
  end for
  reorder the states of OPEN in a non-decreasing order of f-values;
end while

Algorithm 1 is an iterative search in the RG of a place-timed PN. The algorithm utilizes OPEN to store the states that have been generated but not yet expanded, while it uses CLOSED to contain the states that have already been expanded. For each generated state, the algorithm evaluates it with a function $f(S) = g(S) + h(S)$, where $f(S)$ estimates the minimal cost of the transfer from S_0 to S_G along an optimal path that passes through S. Here, $g(S)$ represents the cost already incurred to reach state S from S_0, and $h(S)$ is an estimation of the optimal cost of a transfer from S to S_G. During each iteration, the algorithm selects the state in OPEN with the lowest f-value for expansion, generating its immediate successors. If the goal state S_G is selected for expansion, the search ends and produces a transition firing sequence from S_0 to S_G by tracing the pointers in the states.

3 New Heuristic Function

For an A* search, the choice of the heuristic function is pivotal in influencing the effectiveness of the search, impacting application scenarios, search efficiency, and solution quality. This section first introduces some properties of the heuristic function used by an A* search and then proposes a new heuristic function for the PN-based A* search. It is proven to be admissible and more informed than those in Luo *et al.* (2015), which makes the PN-based A* search more efficient and its obtained result optimal. Additionally, it can deal with timed PNs with multiple resource acquisitions, weighted arcs, redundant resources, and outdated resources, which are commonly seen in the PNs of AMSs.

The heuristic function h used in an A* search mainly possesses the following two critical properties (Edelkamp & Schroedl, 2012; Huang & Zhou, 2020).

3.0.1 Admissibility

h is admissible if $\forall S \in \mathcal{R}(\mathcal{N}, S_0)$, $h(S) \leqslant h^*(S)$ in which $h^*(S)$ represents the minimum cost of a state transfer from S to S_G.

Proposition 1 *An A* search with an admissible h ensures that its obtained results are optimal.*

3.0.2 More-Informedness

Let h_1 and h_2 be two heuristic functions. h_1 is said to be more informed than h_2 if both of them are admissible, $\forall S \in \mathcal{R}(\mathcal{N}, S_0)$, $h_1(S) \geqslant h_2(S)$, and $\exists S' \in \mathcal{R}(\mathcal{N}, S_0)$, $h_1(S') > h_2(S')$.

Note that for an admissible heuristic function h, more-informedness implies closer proximity to h^*.

Proposition 2 *An A* search with a more-informed admissible h has greater pruning power and becomes more efficient.*

In Luo *et al.* (2015), two admissible heuristic functions are formulated for the timed PN-based A* search. The first one calculates the average minimum time (cost) for a resource unit to complete its remaining operations from the current state S to S_G and uses this value as the heuristic. It is detailed as follows.

$$h_{L1}(S) = \frac{\sum\limits_{j(p,i)\in J} (R(p,i) + X(p))}{|ER|} \tag{1}$$

where $j(p,i)$ denotes the ith part token in a non-resource $p \in P_{\widetilde{R}} = P \setminus P_R$ at S, which is a token representing a part (i.e., a component of a product) to be processed, J represents the set of all part tokens, $X(p)$ is the minimum time for an available part token to move from p to its end place, $R(p,i)$ is the remaining token time of $j(p,i)$ at S, and $|ER|$ is the total count of resource units available in the system. Please be aware that a token in $p \in P_{\widetilde{R}}$ is called an available token if its remaining token time is zero, which means it is currently ready for firing a transition $t \in p^\bullet$; otherwise, it is called an unavailable one. The second heuristic function is formulated as below.

$$h_{L2}(S) = \frac{\sum\limits_{j(p,i)\in J} (R(p,i) + X(p)) + \sum\limits_{r\in P_R} \delta(S,r)\cdot G(S,r)}{|ER|} \tag{2}$$

where $G(S,r)$ denotes the minimal idle time of $r \in P_R$ from S to the time when r is used, and $\delta(S,r)$ is a two-valued function. If there is a place $p \in P_{\widetilde{R}} \cap {}^\bullet(r^\bullet)$ satisfying that $M(p) > 0$ and $G(S,r) = \min\{G(S,r')|r' \in P_R \cap {}^\bullet(p^\bullet)\}$, then $\delta(S,r) = 1$; otherwise, $\delta(S,r) = 0$.

Both h_{L1} and h_{L2} are admissible, and h_{L2} is similar to h_{L1} except that h_{L2} includes the sum of the minimal idle time of all resources, which is essential for the transfer from S to S_G. Thus, h_{L2} is more informed than h_{L1}.

The above heuristic functions (Luo *et al.*, 2015) and their deadlock control policies for a timed PN-based A* search are dedicated to revised S^{3}PR nets, which are ordinary PNs where each operation place requires at most a single resource unit. However, in the PN of an AMS, an operation place may need multiple resource units of different types (e.g., Fig. 1(a) shows that p_1 requires two resource units from r_1 and r_2) or the same type (e.g., Fig. 1(b) illustrates that p_1 requires two resource units from r_1). In addition, h_{L1} and h_{L2} consider that all resource units will be used at any reachable state. However, for some reachable states, there may exist some resource units that are never used in the transfer from S to S_G (e.g., Fig. 1(c) shows that the net does not use a resource unit of r_1). Such resource units are called redundant ones. In addition, there also may exist resources that were used but will never be used again (e.g., Fig. 1(d) illustrates that r_1 will not be used again). Such resources are called outdated ones. All those cases are commonly seen in practical AMSs.

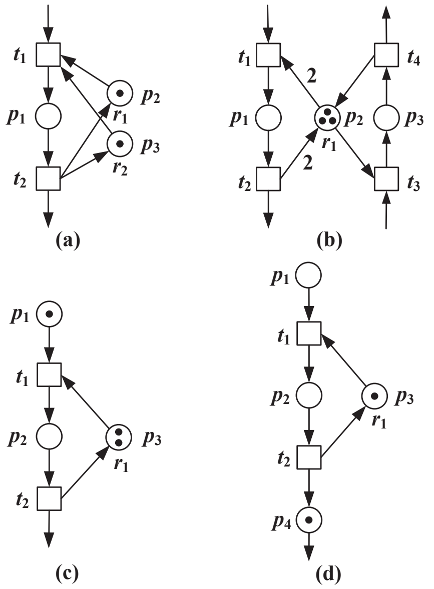

Fig. 1. Four cases to be considered: (a) Multiple resource acquisition; (b) weighted arcs; (c) a redundant resource unit; and (d) an outdated resource.

In the sequel, we propose a new heuristic function that takes into account the above cases of net structures and state behavior and uses them to calculate states' heuristic values for the PN-based A* search. Before introducing the new heuristic function, we define two vectors that will be used: Extended Operation Time (EOT) and Minimal Remaining extended operation Time (MRT). Let $U(p, r)$ be the number of resource units of r that are needed by $p \in P_A$. EOT is a column vector with $|P_A|$ elements such that $\forall p \in P_A$, $\mathrm{EOT}(p)$ is the summation of the operation time in p for each resource unit that is required by p, i.e., $\mathrm{EOT}(p) = D(p) \cdot \sum_{r \in P_R} U(p, r)$. For example, we consider the place-timed PN in Fig. 2, where $P_R = \{p_9, p_{10}\}$ and there exist multiple resource acquisitions, weighted arcs, redundant resource units, and outdated resources. An operation place p_2 requires two resource units which respectively come from r_1 and r_2, and its operation time $D(p_2) = 7$. Thus, $\mathrm{EOT}(p_2) = 7 \times 2 = 14$. Similarly, $\mathrm{EOT}(p_3) = 4 \times 2 = 8$, $\mathrm{EOT}(p_6) = 3 \times 2 = 6$, and $\mathrm{EOT}(p_7) = 2 \times 1 = 2$. MRT (denoted as Φ) is a column vector with $|P_{\widetilde{R}}|$ elements such that $\forall p \in P_{\widetilde{R}}$, $\Phi(p)$ is the minimal total EOT that is required for an available token in p to move from p to its end place. For example, the MRT vector (transposed) of the PN in Fig. 2 is given below.

$$\Phi^T = \begin{bmatrix} 22 & 8 & 0 & 0 & 8 & 2 & 0 & 0 \end{bmatrix}$$

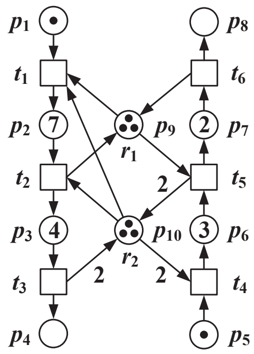

Fig. 2. An example place-timed PN.

Based on them, a new heuristic function h_{ours} for the PN-based A* search is given in Eq.(3) where $C(r)$ is the count of resource units of $r \in P_R$, $P_{\widetilde{R}} = \{M(p) | p \in P_{\widetilde{R}}\}$, and $\mathbf{0}$ is a zero vector. MR^3 (Maximal Remaining Required Resources, denoted as Λ) is a $|P_{\widetilde{R}}| \times |P_R|$ matrix such that if moving an unavailable token in a place $p \in P_{\widetilde{R}}$ from p to its end place requires at most n units of r, $\Lambda(p, r) = n$. $\Lambda(\bullet, r)$ denotes the column vector of Λ that is related to r.

In Eq. (3), its numerator is an addition of two parts. The first part represents the sum of the remaining extended token time and the minimal remaining extended operation time for all part tokens to move from S to S_G. It takes into account multiple resource acquisitions and weighted arcs, which are common in the PNs of AMSs. The second part represents the total idle time of resources like that of Eq. (2). In the denominator of Eq. (3), $M^T_{P_{\widetilde{R}}} \cdot \Lambda(\bullet, r)$ returns the maximal usage count of r in a transfer from S to S_G. Thus, the denominator returns the maximal number of resource units that may be needed in the state transfer, which excludes both redundant resource units and outdated resources at S. Therefore, Eq. (3) represents the average extended time (including remaining extended token time, minimal remaining extended operation time, and resource idle time) for a resource unit that is likely to be used in the state transfer from S to S_G.

To illustrate it, we use the PN in Fig. 2 as an example. $\Lambda(p_3, r_1) = 0$ and $\Lambda(p_3, r_2) = 2$ since moving an unavailable token in p_3 from p_3 to its end place p_4 requires no r_1 and 2 units of r_2. Similarly, we can obtain its MR^3 matrix (transposed) as follows.

$$\Lambda^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 3 & 3 & 2 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}$$

In addition, we have

$$U^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}.$$

$$h_{\text{ours}}(S)=\begin{cases}\dfrac{\sum\limits_{j(p,i)\in J}\left(R(p,i)\cdot\sum\limits_{r\in P_R}U(p,r)+\Phi(p)\right)+\sum\limits_{r\in P_R}\delta(S,r)\cdot G(S,r)}{\sum\limits_{r\in P_R}\min\left\{M^T_{P_{\widetilde{R}}}\cdot\Lambda(\bullet,r),\ C(r)\right\}}, & \text{if } M^T_{P_{\widetilde{R}}}\cdot\Lambda\neq\mathbf{0}\\ 0, \text{ otherwise}\end{cases} \tag{3}$$

Let S_G represent the state in which all part tokens have reached their respective end places. Consider a state $S=(M,R)$ with

$$M^T=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2\end{bmatrix}$$

and

$$R^T=\begin{bmatrix}0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}.$$

Since $\sum_{r\in P_R}U(p_2,r)=2$, $\sum_{r\in P_R}U(p_7,r)=1$, $\Phi(p_2)=8$, $\Phi(p_7)=0$, $\delta(S,r_1)=0$, $\delta(S,r_2)=1$, $G(S,r_2)=3$, $M^T_{P_{\widetilde{R}}}\cdot\Lambda(\bullet,r_1)=2$, $M^T_{P_{\widetilde{R}}}\cdot\Lambda(\bullet,r_2)=3$, and $C(r_1)=C(r_2)=3$, we obtain that $h_{\text{ours}}(S)=\frac{(3\times2+8+1\times1+0)+0+1\times3}{2+3}=3.6$. Consider another state $S'=(M',R')$ with

$$(M')^T=\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 3 & 1\end{bmatrix}$$

and

$$(R')^T=\begin{bmatrix}0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}.$$

Since $\sum_{r\in P_R}U(p_3,r)=2$, $\sum_{r\in P_R}U(p_8,r)=0$, $\Phi(p_3)=\Phi(p_8)=0$, $\delta(S',r_1)=\delta(S',r_2)=0$, $M^T_{P_{\widetilde{R}}}\cdot\Lambda(\bullet,r_1)=0$, and $M^T_{P_{\widetilde{R}}}\cdot\Lambda(\bullet,r_2)=2$, we have $h_{\text{ours}}(S')=\frac{(4\times2+0+0\times0+0)+0+0}{0+2}=4$.

The new heuristic function utilizes a timed PN's multiple resource acquisitions (e.g., in Fig. 2, p_2 requires different resources from r_1 and r_2), weighted arcs (e.g., both p_3 and p_6 require multiple units from r_2), redundant resource units for a state (e.g., a unit of r_1 and a unit of r_2 are never required in the transfer from S to S_G and the transfer from S' to S_G, respectively), and outdated resources for a state (e.g., r_1 is no longer required in the state transfer from S' to S_G) to estimate the optimal time or cost for the transfer from S to S_G. In addition, h_{ours} is admissible, which guarantees its result's optimality, and it is more informed than both h_{L1} and h_{L2} if the modeled PN has more than one resource to make the A* search faster.

Theorem 1 *h_{ours} is admissible.*

Proof. *According to Eq. (3), $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, $h_{ours}(S)$ denotes the average remaining extended time of each resource unit that is required in the transfer from S to S_G. This calculation assumes that all the required resource units are used concurrently, and each part token will follow the path with the minimum remaining extended operation time. Actually, resource units are often not concurrently used. In addition, in an optimal state transfer from S to S_G, some part tokens may not move along a path with the minimal remaining extended operation time. Thus, $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, $h_{ours}(S)$ serves as a lower bound on the time required for the PN to transfer from S to S_G, i.e., $h_{ours}(S)\leqslant h^*(S)$. Therefore, h_{ours} is admissible.*

Therefore, according to Proposition 1, Algorithm 1 with h_{ours} ensures that its obtained schedules are optimal.

Theorem 2 *h_{ours} is more informed than both h_{L1} and h_{L2} if $|P_R|\geqslant 2$.*

Proof. Since h_{L2} is more informed than h_{L1}, we only need to prove that h_{ours} is more informed than h_{L2} if $|P_R|\geqslant 2$.

First, we already know that both h_{ours} and h_{L2} are admissible according to Theorem 1 and (Luo *et al.*, 2015), respectively. Next, we show that $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, $h_{\text{ours}}(S)\geqslant h_{L2}(S)$. The numerator of $h_{\text{ours}}(S)$, i.e., the addition of remaining extended token time, minimal remaining extended operation time, and resource idle time, considers the multiple usages of resource units by respectively multiplying remaining token time and operation time by the number of resource units used by an operation, which is equal to or greater than one. Thus, the numerator of $h_{\text{ours}}(S)$ is not less than that of $h_{L2}(S)$, which does not consider multiple resource acquisitions of an operation place. That is to say, $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, the numerator of $h_{\text{ours}}(S)$ is not less than that of $h_{L2}(S)$. On the other hand, $|ER|$ is the overall quantity of resource units in the system, and the denominator of $h_{\text{ours}}(S)$ only counts the resource units to be used in the transfer from S to S_G. So, $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, the denominator of $h_{\text{ours}}(S)$ is not greater than that of $h_{L2}(S)$. Thus, $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, $h_{\text{ours}}(S)\geqslant h_{L2}(S)$ holds.

Finally, we prove that if $|P_R|\geqslant 2$, $\exists S'\in\mathcal{R}(\mathcal{N},S_0)$ such that $h_{\text{ours}}(S')>h_{L2}(S')$. $\forall S\in\mathcal{R}(\mathcal{N},S_0)$, if there exists a path from S to S_G where all part tokens are in their end places, the path must have a state S' such that all part tokens reach their end places except that one part token is in its last operation place. Then, there are only two cases: 1) This part token requires only one resource to finish its operation. In this case, the numerator of $h_{\text{ours}}(S')$ is not less than that of $h_{L2}(S')$ since the part token needs one or more resource units of the same type, while the denominator of $h_{\text{ours}}(S')$ is less than that of $h_{L2}(S')$ since $|P_R| \geqslant 2$, which implies some resources must be outdated. 2) This part token requires multiple resources to finish its operation. In such a case, the numerator of $h_{\text{ours}}(S')$ is greater than that of $h_{L2}(S')$ owing to multiple resource acquisitions of the operation, while the denominator of $h_{\text{ours}}(S')$ is not greater than that of $h_{L2}(S')$. Thus, if $|P_R| \geqslant 2$, $\exists S' \in \mathcal{R}(\mathcal{N}, S_0)$ such that $h_{\text{ours}}(S') > h_{L2}(S')$.

Since 1) both h_{ours} and h_{L2} are admissible, 2) $\forall S \in \mathcal{R}(\mathcal{N}, S_0)$, $h_{\text{ours}}(S) \geqslant h_{L2}(S)$, and 3) if $|P_R| \geqslant 2$, $\exists S' \in \mathcal{R}(\mathcal{N}, S_0)$, $h_{\text{ours}}(S') > h_{L2}(S')$, we have that h_{ours} is more informed than h_{L2} if $|P_R| \geqslant 2$.

Therefore, according to Proposition 2, h_{ours} leads to a more efficient A* search than both h_{L1} and h_{L2} if $|P_R| \geqslant 2$.

4 Illustrative Examples

In this section, we conducted tests on several commonly used AMS benchmark systems using the timed PN-based A* search algorithm with our heuristic function and other comparable alternatives. The experiments were performed using C# programs on a Windows computer equipped with an Intel i5 Core 2.20GHz CPU and 12GB of memory.

First, we consider the PN model illustrated in Figure 2. The model consists of six transitions and ten places, where $P_S = \{p_1, p_5\}$, $P_E = \{p_4, p_8\}$, $P_R = \{p_9, p_{10}\}$, and the remaining places belong to P_A. We conducted seven tests, each of which has a different lot size in P_S, by using Algorithm 1 with a non-informed evaluation function $h = 0$, an informed heuristic function h_{L2}, and ours, respectively. Table 1 shows their results, including the obtained schedule's makespan ($\mathcal{M}$), the number of expanded states (N_E), and the algorithm's search time ($\mathcal{T}$).

From Table 1, we can see that for each lot size, the timed PN-based A* obtains the schedules with the same makespan regardless of using $h = 0$, h_{L2}, or h_{ours} since they are all admissible, which ensures that their obtained schedules are optimal according to Proposition 1. In addition, the two informed heuristic functions, h_{L2} and h_{ours}, make the PN-based A* search more efficient than the non-informed one $h = 0$. More importantly, since h_{ours} is more informed than h_{L2}, it makes the PN-based A* search expand fewer states than h_{L2} for all cases. Note that for this simple example PN with small lot sizes, the PN-based A* search with h_{ours} expanded fewer states but may need a little more computational time than that with h_{L2}. This is probably due to its extra initialization time of some vectors and matrices, such as the MRT vector and the MR^3 matrix. As the lot size increases, we can see that for our method, the time saved by expanding fewer states will surpass the vector and matrix initialization time.

Then, we consider a bigger generalized place-timed PN of an AMS adapted from Huang & Zhou (2020). Its PN model is shown in Fig. 3, which comprises 20 transitions and 26 places where $P_S = \{p_1, p_5, p_{14}\}$, $P_E = \{p_{27} - p_{29}\}$, $P_R = \{p_{20} - p_{26}\}$, and the remaining places form P_A. Four distinct lot sizes within P_S are tested by using the PN-based A* search with $h = 0$, h_{L2}, and h_{ours}, respectively. Table 2 shows their results. From the table, we have that the PN-based A* search with the proposed heuristic function h_{ours} expands fewer states and needs less search time than those with $h = 0$ or h_{L2} for all cases. Additionally, according to Proposition 1, the obtained schedules are ensured to be optimal since h_{ours} is admissible.

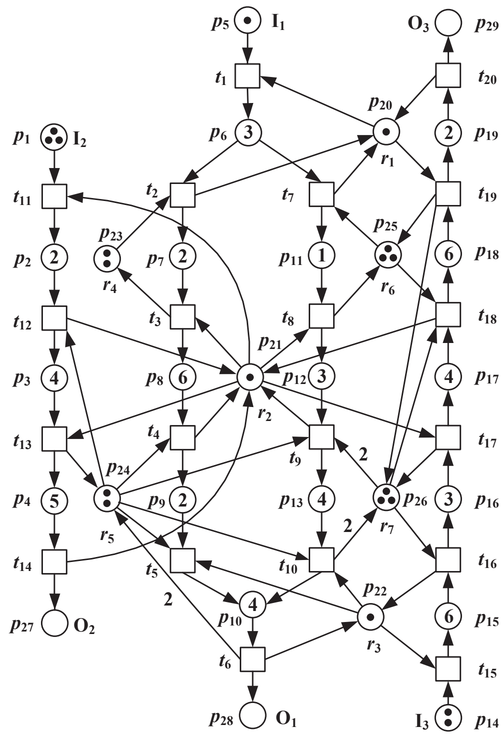

Fig. 3. A generalized timed PN of an AMS adapted from Huang & Zhou (2020).

Finally, we consider another bigger generalized place-timed PN of an AMS adapted from Huang *et al.* (2022). Its place-timed PN is depicted in Fig. 4, which contains 38 transitions and 40 places in which $P_S = \{p_1, p_{12}, p_{21}, p_{31}\}$, $P_E = \{p_{11}, p_{20}, p_{30}, p_{37}\}$, $P_R = \{p_{38} - p_{40}\}$, and the rest of the places belong to P_A. It is also tested by the PN-base A* with $h = 0$, h_{L2}, and

Table 1
Scheduling Comparisons for the timed PN in Fig. 2

p_1	p_5	PN-based A* with $h = 0$			PN-based A* with h_{L2}			PN-based A* with h_{ours}		
		$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$
1	1	11	17	5.81s	11	15	5.86s	11	13	6.27s
2	2	17	192	6.16s	17	179	6.60s	17	150	6.34s
3	3	24	696	7.34s	24	684	7.19s	24	595	7.11s
4	4	31	1509	9.22s	31	1489	8.94s	31	1376	8.86s
5	5	38	2605	12.61s	38	2583	11.59s	38	2453	11.57s
6	6	45	3982	16.59s	45	3959	15.99s	45	3826	15.86s
10	10	73	12330	72.82s	73	12320	69.94s	73	12144	65.32s

Table 2
Scheduling Comparisons for the timed PN in Fig. 3

p_1	p_5	p_{14}	PN-based A* with $h = 0$			PN-based A* with h_{L2}			PN-based A* with h_{ours}		
			$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$
1	1	1	21	1237	6.26s	21	1041	5.29s	21	819	4.80s
2	1	1	26	4542	22.85s	26	4450	21.89s	26	3587	17.71s
2	2	1	29	35454	548.38s	29	35151	544.66s	29	30491	457.67s
2	2	2	33	239071	22644.58s	33	236813	21990.42s	33	218475	20363.82s

h_{ours}, respectively. Its results are shown in Table 3, which shows that the PN-based A* with $h = 0$ cannot solve it in an acceptable time, but those with h_{L2} and h_{ours} can solve it by expanding 87,254 states in 5,702 seconds and by expanding 64,350 states in about one hour, respectively. Both the number of expanded states and computational time of the PN-based A* search with h_{ours} are much less than those of the A* with h_{L2}. Table 4 presents our obtained schedule, which is a transition firing path that stands for a sequence of executions within the underlying system. According to Proposition 1, it is ensured to be an optimal schedule having the lowest cost since h_{ours} is admissible.

Table 4
The Obtained Optimal Schedule for the Timed PN in Fig. 4 (Makespan = 350).

t	Firing Time	t	Firing Time	t	Firing Time
t_1	0	t_5	153	t_{21}	258
t_{23}	0	t_{26}	153	t_{38}	268
t_{33}	0	t_{14}	173	t_{31}	268
t_2	69	t_{15}	173	t_6	268
t_{13}	78	t_{36}	175	t_7	268
t_{24}	78	t_{37}	175	t_{32}	338
t_{25}	78	t_{29}	175	t_8	350
t_{34}	99	t_{30}	231	t_{22}	350
t_{35}	99	t_{16}	258		

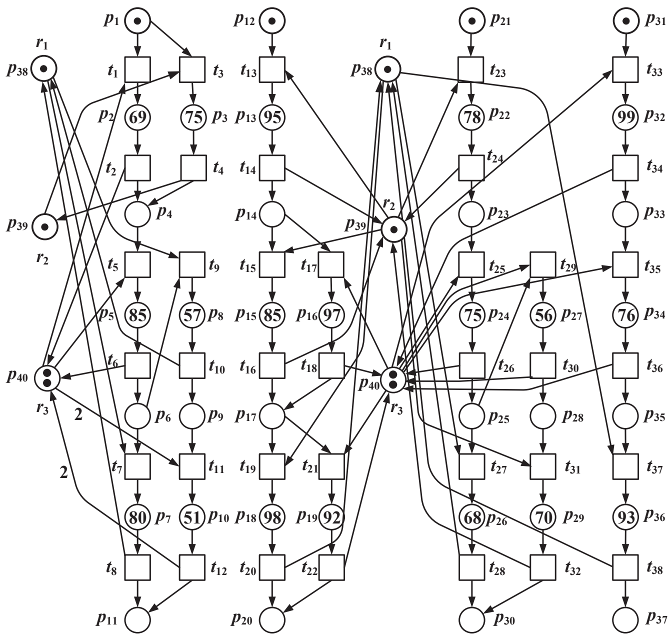

Fig. 4. Another generalized timed PN of an AMS adapted from Huang *et al.* (2022).

5 Discussions

The proposed method is more general and efficient than that presented in Luo *et al.* (2015). However, similar to the approach in Luo *et al.* (2015), it is also afflicted by the state explosion problem when it tries to obtain optimal results, primarily because of the NP-hard nature of the scheduling problem, making it impractical for application in large and intricate cases. To alleviate this limitation, several existing techniques show promise if

Table 3
Scheduling Comparisons for the timed PN in Fig. 4

p_1	p_{12}	p_{21}	p_{31}	PN-based A* with $h=0$			PN-based A* with h_{L2}			PN-based A* with h_{ours}		
				$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$	$\mathcal{M}$	N_E	$\mathcal{T}$
1	1	1	1	-	-	-	350	87254	5702.54s	350	64350	3564.14s

"-" means it cannot terminate in ten hours.

combined with our method for scheduling more complex cases.

The first category of the methods involves applying transformations to PNs before constructing or searching their RGs. For instance, a net reduction method is proposed in Uzam (2004), aimed at simplifying models for streamlined calculations.

The second category is symmetry-based methods. Many systems have some form of symmetry, for example, having identical components interconnected in a regular manner. The idea behind state space reduction via symmetry is to store just one representative state from a group of equivalent states (Jensen & Kristensen, 2009).

To accelerate the search process, binary decision diagrams have been employed to compactly represent state sets and efficiently execute set operations (Huang & Zhou, 2022), and an anytime search scheme is adopted to generate a feasible schedule quickly and then continuously improve the result if given more computational time (Lv & Huang, 2024).

Furthermore, various algorithms combine PNs with meta-heuristics (Huang *et al.*, 2023), such as genetic algorithms, ant colony optimization, particle swarm optimization, and hybrid methods combining multiple strategies to obtain feasible solutions. These algorithms typically treat transition firing sequences as candidate solutions, generating feasible solutions. However, they are usually offline algorithms. Therefore, designing efficient and high-quality PN-based meta-heuristic methods to accelerate the scheduling process is critical.

Additionally, machine learning (ML) emerges as a promising approach for real-time scheduling problems, capable of learning from training data and making rapid classifications or predictions. ML methods, including supervised learning, unsupervised learning, and reinforcement learning based on available feedback, are increasingly applied across various domains, including game theory, control theory, multi-agent systems, and operations research. As an example, reinforcement learning has demonstrated successful applications in system scheduling, including the prediction of agent actions (Zhang *et al.*, 2022), the evolution of genetic programming rules (Gu *et al.*, 2022), and the optimization of dispatching rules (Priore *et al.*, 2018).

Moreover, non-search-based scheduling methods based on PNs also exist. In Wu (1999); Wu & Zhou (2001), resource-oriented PNs (ROPNs) are proposed to model and schedule discrete event systems. In ROPNs, processes are represented as components sequentially interacting with their respective resources, and color coding is introduced to differentiate between different component flows. This method facilitates the obtaining of optimal solutions in a closed-form manner, without constructing or searching in the PN's RG. This approach has achieved success in scheduling cluster tools for semiconductor manufacturing (Wu *et al.*, 2008*b*; Pan *et al.*, 2015; Qiao *et al.*, 2015; Zhu *et al.*, 2018; Yang *et al.*, 2018, 2020). Additionally, it has been extended to schedule oil refineries using hybrid PNs (Wu *et al.*, 2008*a*). For further information, please consult (Pan *et al.*, 2018).

6 Conclusion

The selection of an appropriate heuristic function in an A* search algorithm holds significant importance, as it directly impacts its applicability in various scenarios, as well as the quality of results and search efficiency achieved. The paper introduces a novel heuristic function for the A* search based on place-timed PNs. When compared with those in Luo *et al.* (2015), it can deal with multiple resource acquisitions, weighted arcs, redundant resource units, and outdated resources, which are common in the PNs of AMSs, and it is proven admissible and more informed, which ensures that its obtained schedule is optimal and its PN-based A* search expands fewer states and searches more efficiently.

References

Edelkamp, S., & Stefan Schroedl (2012). *Heuristic Search: Theory and Applications*. Elsevier. Waltham, MA, USA.

Gu, W., Yuxin Li, Dunbing Tang, Xianliang Wang, & Minghai Yuan (2022). 'Using real-time manufacturing data to schedule a smart factory via reinforcement learning'. *Computers and Industrial Engineering* **171**, 108406.

Huang, B., & MengChu Zhou (2020). *Supervisory Control and Scheduling of Resource Allocation Systems: Reachability Graph Perspective*. John Wiley & Sons. Hoboken, NJ, USA.

Huang, B., & MengChu Zhou (2022). 'Symbolic scheduling of robotic cellular manufacturing systems with timed Petri nets'. *IEEE Transactions on Control Systems Technology* **30**(5), 1876–1887.

Huang, B., MengChu Zhou, Abdullah Abusorrah, & Khaled Sedraoui (2022). 'Scheduling robotic cellular manufacturing systems with timed Petri net, A* search, and admissible heuristic function'. *IEEE Transactions on Automation Science and Engineering* **19**(1), 243–250.

Huang, B., Mengchu Zhou, Xiaoyu Sean Lu, & Abdullah Abusorrah (2023). 'Scheduling of resource allocation systems with timed Petri nets: A Survey'. *ACM Computing Surveys* **55**(11), 1–27.

Jensen, K., & Lars M Kristensen (2009). *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media. Berlin, Germany.

Lee, D. Y., & Frank DiCesare (1994). 'Scheduling flexible manufacturing systems using Petri nets and heuristic search'. *IEEE Transactions on Robotics and Automation* **10**(2), 123–132.

Luo, J., KeYi Xing, MengChu Zhou, XiaoLing Li, & XinNian Wang (2015). 'Deadlock-free scheduling of automated manufacturing systems using Petri nets and hybrid heuristic search'. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **45**(3), 530–541.

Lv, J., & Bo Huang (2024). 'A Petri-net-based anytime A* search for scheduling resource allocation systems'. *IEEE Transactions on Industrial Informatics* **20**(2), 2865–2872.

Mejía, G., & Jordi Pereira (2020). 'Multiobjective scheduling algorithm for flexible manufacturing systems with Petri nets'. *Journal of Manufacturing Systems* **54**, 272–284.

Mejía, G., Karen Niño, Carlos Montoya, María Angélica Sánchez, Jorge Palacios, & Lionel Amodeo (2016). 'A Petri net-based framework for realistic project management and scheduling: An application in animation and videogames'. *Computers & Operations Research* **66**, 190–198.

Pan, C., MengChu Zhou, Yan Qiao, & NaiQi Wu (2018). 'Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges'. *IEEE Transactions on Automation Science and Engineering* **15**(2), 586–601.

Pan, C., Yan Qiao, MengChu Zhou, & NaiQi Wu (2015). 'Scheduling and analysis of start-up transient processes for dual-arm cluster tools with wafer revisiting'. *IEEE Transactions on Semiconductor Manufacturing* **28**(2), 160–170.

Pinto, M., Cristóvão Silva, Matthias Thürer, & Samuel Moniz (2024). 'Nesting and scheduling optimization of additive manufacturing systems: Mapping the territory'. *Computers & Operations Research* p. 106592.

Priore, P., Borja Ponte, Javier Puente, & Alberto Gómez (2018). 'Learning-based scheduling of flexible manufacturing systems using ensemble methods'. *Computers and Industrial Engineering* **126**, 282–291.

Qiao, Y., NaiQi Wu, QingHua Zhu, & LiPing Bai (2015). 'Cycle time analysis of dual-arm cluster tools for wafer fabrication processes with multiple wafer revisiting times'. *Computers & Operations Research* **53**, 252–260.

Uzam, M. (2004). 'The use of the Petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems'. *International Journal of Advanced Manufacturing Technology* **23**(3-4), 204–219.

Uzam, M., Zhiwu Li, & Umar Suleiman Abubakar (2016). 'Think globally act locally approach for the synthesis of a liveness-enforcing supervisor of FMSs based on Petri nets'. *International Journal of Production Research* **54**(15), 4634–4657.

Wu, N. (1999). 'Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored Petri net model'. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **29**(2), 192–204.

Wu, N., & MengChu Zhou (2001). 'Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems'. *IEEE Transactions on Robotics and Automation* **17**(5), 658–669.

Wu, N., Feng Chu, ChengBin Chu, & MengChu Zhou (2008*a*). 'Short-term schedulability analysis of crude oil operations in refinery with oil residency time constraint using Petri nets'. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **38**(6), 765–778.

Wu, N., MengChu Zhou, & Feng Chu (2008*b*). 'A Petri net-based heuristic algorithm for realizability of target refining schedule for oil refinery'. *IEEE Transactions on Automation Science and Engineering* **5**(4), 661–676.

Yang, F., Naiqi Wu, Yan Qiao, & Rong Su (2018). 'Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets'. *IEEE/CAA Journal of Automatica Sinica* **5**(1), 270–280.

Yang, F., NaiQi Wu, Yan Qiao, MengChu Zhou, Rong Su, & Ting Qu (2020). 'Modeling and optimal cyclic scheduling of time-constrained single-robot-arm cluster tools via Petri nets and linear programming'. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **20**(3), 871–883.

Zhang, Y., Haihua Zhu, Dunbing Tang, Tong Zhou, & Yong Gui (2022). 'Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems'. *Robotics and Computer-Integrated Manufacturing* **78**, 102412.

Zhu, Q., MengChu Zhou, Yan Qiao, & NaiQi Wu (2018). 'Petri net modeling and scheduling of a close-down process for time-constrained single-arm cluster tools'. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **48**(3), 389–400.