Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Parallel fault diagnosis using hierarchical fuzzy Petri net by reversible and dynamic decomposition mechanism*

Yinhong XIANG[1], Kaiqing ZHOU‡[1], Arezoo SARKHEYLI-HÄGELE[2], Yusliza YUSOFF[3], Diwen KANG[1], Azlan Mohd ZAIN[3]

[1]*School of Communication and Electronics Engineering, Jishou University, Jishou 416000, China*
[2]*Internet of Things and People Research Center, Department of Computer Science and Media Technology, Malmö University, Malmö 20506, Sweden*
[3]*Faculty of Computing, Universiti Teknologi Malaysia, Skudai 81310, Malaysia*
E-mail: yhxiang@stu.jsu.edu.cn; kqzhou@jsu.edu.cn; arezoo.sarkheyli-haegele@mau.se; yusliza@utm.my; kangdiwen@jsu.edu.cn; alzanmz@utm.my

Received Mar. 12, 2024; Revision accepted Apr. 1, 2024; Crosschecked Oct. 12, 2024; Published online Dec. 27, 2024

Abstract: The state space explosion, a challenge analogous to that encountered in a Petri net (PN), has constrained the extensive study of fuzzy Petri nets (FPNs). Current reasoning algorithms employing FPNs, which operate through forward, backward, and bidirectional mechanisms, are examined. These algorithms streamline the inference process by eliminating irrelevant components of the FPN. However, as the scale of the FPN grows, the complexity of these algorithms escalates sharply, posing a significant challenge for practical applications. To address the state explosion issue, this work introduces a parallel bidirectional reasoning algorithm for an FPN that utilizes reverse and decomposition strategies to optimize the implementation process. The algorithm involves hierarchically dividing a large-scale FPN into two sub-FPNs, followed by a converse operation to generate the reversal sub-FPN for the right-sub-FPN. The detailed mapping between the original and reversed FPNs is thoroughly discussed. Parallel reasoning operations are then conducted on the left-sub-FPN and the resulting reversal right-sub-FPN, with the final result derived by computing the Euclidean distance between the outcomes from the output places of the two sub-FPNs. A case study is presented to illustrate the implementation process, demonstrating the algorithm's significant enhancement of inference efficiency and substantial reduction in execution time.

Key words: Fuzzy Petri net (FPN); State explosion; Decomposition; Parallel; Bidirectional reasoning
https://doi.org/10.1631/FITEE.2400184 **CLC number:** TP301

1 Introduction

With the complexity of industrial manufacturing equipment and production processes increasing dramatically, a correspondingly extensive array of fault diagnosis methods has been proposed and discussed. These include the application of Petri nets (PNs) (Liu et al., 2017), deep learning techniques (Lei et al., 2020; Chen XH et al., 2021), and swarm intelligence algorithms (Ziani et al., 2017; Chen RH et al., 2020; Ye et al., 2023). As a distinctive tool within the knowledge-driven approach, the fuzzy Petri net (FPN) has been

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 62066016), the Natural Science Foundation of Hunan Province of China (No. 2023JJ2279), the Scientific Research Project of Education Department of Hunan Province of China (Nos. 22B0549 and 22C0282), the Postgraduate Scientific Research Innovation Project of Hunan Province (No. CX20231088), the Fundamental Research Grant Scheme of Malaysia (No. R.J130000.7809.5F524), and the UTMFR Grant Research Management Center (RMC) of Universiti Teknologi Malaysia (UTM) (No. Q.J130000.2551.20H71)
ORCID: Yinhong XIANG, https://orcid.org/0009-0008-7629-868X; Kaiqing ZHOU, https://orcid.org/0000-0001-5779-7135

widely utilized for modeling, simulating, and executing diagnostic tasks within complex manufacturing systems, leading to fruitful results. This is primarily attributable to two key characteristics of FPNs (Zhou and Zain, 2016; Seatzu, 2019; Liu et al., 2022; Wang et al., 2022). First, the FPN maintains the PN's ability to describe asynchronous concurrency and provides a graphical representation. Second, it offers a formal modeling approach for managing fuzzy and uncertain information within knowledge-based systems. This sets FPN apart from the opacity and black-box nature of deep learning, as FPNs can explicitly describe the states, events, and transitions within the system, thereby enhancing the interpretability and explainability of the inference process (Rudin, 2019). Recently, more high-level PNs with fuzzy factors have been introduced to represent knowledge effectively and accommodate the diverse constraints inherent in real engineering issues. Notable examples include probabilistic linguistic PNs (Shi et al., 2024), Z-number PNs (Shi et al., 2022), spherical linguistic PNs (Mou et al., 2022), and linguistic PNs (Liu et al., 2022).

As knowledge-based systems increase in size, their corresponding FPNs grow in scale (Zhou et al., 2019). This scaling phenomenon, known as the state explosion problem, poses a substantial challenge to the construction and application of FPNs (Valmari, 1998; Grobelna and Karatkevich, 2021). To address the state explosion problem, researchers have developed a suite of decomposition algorithms and reasoning techniques. These methods ensure that the resulting sub-PNs preserve the consistency of the original PN, thereby reducing the model's scale. Chen SM (2000) proposed a fuzzy "AND/OR" graph via FPN backward reasoning, which constructs an "AND/OR" graph from target places and reduces the model size. This approach also allows for the flexible evaluation of the truth degree for any user-specified proposition. Zaitsev (2004) discussed functional sub-PNs based on structural properties and presented a decomposition algorithm of polynomial complexity $O(n^3)$ that maintains the properties of the original PN. Lakos and Petrucci (2011) introduced a modular PN with dynamic priorities, considering transitions and the current marking rather than only the firing mode. Their approach modularizes the state space, thereby solving the state space explosion problem. Zeng et al. (2012) employed time-scale decomposition to reduce the state space in continuous-time Markov chains and decomposed a stochastic PN into several sub-level models with consistent transition rates. Liu et al. (2013) introduced a fault diagnosis and cause analysis model that uses fuzzy evidence inference within a dynamic adaptive FPN. The model achieves reverse cause analysis through a reverse operation and enhances fault diagnosis accuracy by integrating fuzzy evidence inference into the FPN framework. Nishi and Matsumoto (2015) proposed a method for decomposing large-scale PNs into multiple sub-PNs capable of capturing the behavior of each system component. This method reduces the computational complexity of discovering near-optimal scheduling solutions, enhances solution efficiency, guarantees the prevention of deadlocks, and facilitates the achievement of acyclic scheduling. Shen et al. (2013) proposed a method for solving the state explosion problem of PN using matching theory. This method reduces the PN by fusing transitions with maximum weight values and eliminating redundant positions, resulting in a compressed and reduced PN. Salum (2015) introduced a novel PN analysis tool, the superposition chain, which employs superposition operators to define triggering semantics. This allows transitions to be fired in any order through superpositions, enabling the superposition chain to define an acyclic PN that can be checked in parallel or via superpositions, thus avoiding state explosion and enhancing the efficiency of PN analysis. Zhou et al. (2015a) discussed a bidirectional decomposition algorithm to divide a large-scale FPN into a series of sub-FPNs. However, the practical application of this approach is challenged by the difficulty of implementing inference on each sub-FPN. Subsequently, they developed a bidirectional adaptive inference algorithm based on intrinsic inference paths (Zhou et al., 2018) to overcome the state explosion problem by managing the dimensions of the operation matrix. Despite this advancement, the proposed bidirectional reasoning still predominantly operates with forward inference on the original FPN and backward reasoning on the simplified FPN, which means that it does not employ an accurate parallel reasoning mechanism. Bai et al. (2023) introduced a decomposition algorithm for the q-rung orthopair fuzzy reversed PN using place and transition

vectors. This algorithm was able to avoid interference from unrelated propositions and decrease the complexity of inference. Yang (2023) addressed the state explosion issue of PN by proposing a polynomial decomposition algorithm via P-invariants. This algorithm decomposes complex workflow network models into simple subnet classes, which were effective for describing the process of business case handling.

Researchers focusing on inference in large-scale FPNs commonly employ various mechanisms to simplify the original FPN and enhance reasoning efficiency. These mechanisms include using forward or backward search to identify the most straightforward reasoning path between input and output places (Jiang et al., 2022). The core of this approach is the individual implementation of forward and backward inference mechanisms. Although backward reasoning from output places can effectively capture inner inference paths and directly reduce the complexity of the matrices involved in the forward reasoning operation, the overall complexity remains substantial. In contrast, actual bidirectional reasoning involves executing simultaneous forward and backward reasoning—one starting from the initial state and the other from the goal state—with the expectation that they will converge in the middle (Russell and Norvig, 2009). However, at the current stage, the bidirectional reasoning approach fails to meet the demand for parallelism. Thus, it can be considered a hybrid reasoning technique that integrates forward and backward reasoning. Moreover, when evaluated from a parallel perspective in knowledge reasoning, the existing FPN inference method does not comply with the criteria for a 'true' bidirectional reasoning method.

Building on the above findings, this paper introduces a parallel bidirectional reasoning approach utilizing a hierarchical FPN enabled by a reversible and dynamic decomposition mechanism (PBR-HFPN-RDD). This method aims to achieve parallel fault diagnosis and mitigate the state explosion issues encountered in the following aspects:

1. By analyzing the clear mapping relationship between an FPN and its reverse (Re-FPN) elements, this paper delves into the Re-FPN generation algorithm and associated concepts in depth.

2. By exploring potential reasoning error sources, this paper develops a dynamic decomposition algorithm to partition two sub-FPNs of varying scales by leveraging the properties of the incidence matrix.

3. The PBR-HFPN-RDD is applied to conduct parallel reasoning operations on the derived left-sub-FPN and the Re-FPN, utilizing the obtained right-sub-FPN to achieve the final diagnosis result.

Furthermore, a case study is conducted to validate the correctness of the proposed PBR-HFPN-RDD. The experimental results indicate that PBR-HFPN-RDD can diminish the scale of matrix operations within the FPN reasoning process, thereby mitigating the state explosion issue and enhancing the efficiency of model reasoning.

The remainder of this paper is structured as follows. Section 2 revisits the relevant concepts, including FPN, fuzzy production rules (FPRs), formal inference mechanisms, and the properties of the incidence matrix. Section 3 investigates the mapping relationship between the original FPN and its corresponding Re-FPN and introduces the algorithm for generating the Re-FPN. Section 4 analyzes the potential sources of reasoning error in a parallel fault diagnosis algorithm utilizing the Re-FPN and discusses a dynamic decomposition algorithm that segments the entire FPN, along with a detailed presentation of the PBR-HFPN-RDD. Section 5 demonstrates the feasibility and correctness of the proposed PBR-HFPN-RDD through a case study. Finally, Section 6 provides a summary of the entire paper.

2 Related notions

This section reviews the fundamental definitions of an FPN, an FPR, and their related notions based on the references (Yeung and Ysang, 1998; Chen SM, 2002; Zhou et al., 2015b; Yu et al., 2023).

2.1 Fuzzy Petri net

Definition 1 (FPN) An FPN can be defined as a 9-tuple.

FPN(Σ)=(P, T, F, $\boldsymbol{M}$, $\boldsymbol{W}$, **Th**, **CF**, $\boldsymbol{I}$, $\boldsymbol{O}$), where:

(1) $P=\{p_1, p_2, \cdots, p_n\}$ is a finite set of places;

(2) $T=\{t_1, t_2, \cdots, t_m\}$ is a finite set of transitions;

(3) $F\subseteq(P\times T)\cup(T\times P)$ is a set of directed arcs representing the flow relations between P and T;

(4) $\boldsymbol{M}=(m_1, m_2, \cdots, m_n)^{\mathrm{T}}$ is a vector to record the truth degree of the corresponding place p_i (i=1, 2, $\cdots$, n) and $m_i\in[0,1]$. The initial marking is denoted by $\boldsymbol{M}_0$;

(5) $\boldsymbol{W}=(w_{(i,j)})_{n\times m}$ is a matrix to record the weight from p_i to t_j, $w_{(i,j)}\in(0,1]$;

(6) $\mathbf{Th}=(\mu_1, \mu_2, \cdots, \mu_m)^{\mathrm{T}}$ is a vector to record each transition's threshold t_j (j=1, 2, ⋯, m) and $\mu_j\in(0,1]$;

(7) $\mathbf{CF}=(\mathrm{CF}_{ij})_{n\times m}$ is a matrix to record each certainty factor from t_j to p_i, $\mathrm{CF}_{ij}\in(0,1]$ (i=1, 2, ⋯, n; j=1, 2, ⋯ , m);

(8) $\boldsymbol{I}=(I(p_i, t_j))_{n\times m}$ is an input matrix, where $I(p_i, t_j)$ (i=1, 2, ⋯, n; j=1, 2, ⋯, m) represents a directed arc exists from p_i to t_j and $I(p_i, t_j)=w_{(i,j)}$; otherwise, $I(p_i, t_j)=0$;

(9) $\boldsymbol{O}=(O(p_i, t_j))_{n\times m}$ is an output matrix, where $O(p_i, t_j)$ represents a directed arc exists from t_j to p_i and $O(p_i, t_j)=\mathrm{CF}_{ij}$ (i=1, 2, ⋯, n; j=1, 2, ⋯, m); otherwise, $O(p_i, t_j)=0$.

Definition 2 (Pre-set and post-set) For an FPN, ${}^{\bullet}x=\{y|(y, x)\in F\}$ is the pre-set of x, and $x^{\bullet}=\{y|(x, y)\in F\}$ is the post-set of x (x, $y\in(P\cup T)$).

Definition 3 (Input place and output place) Input place is a set of places that fulfill $\{p\in P|{}^{\bullet}p=\varnothing$ and $p^{\bullet}\neq\varnothing\}$; output place is a set of places that fulfill $\{p\in P|{}^{\bullet}p\neq\varnothing$ and $p^{\bullet}=\varnothing\}$.

Definition 4 (Enable and fire) For a given FPN with a marking $\boldsymbol{M}$, the transition t enables, denoted as $\boldsymbol{M}]>$, once the property that all input places $p\in{}^{\bullet}t$ fulfills the following condition:

$$\sum \boldsymbol{M}(p_i)\times W(i,j)\geq\mu(t_j). \tag{1}$$

Once, and only if, a transition is enabled, the transition fires and produces a new marking $\boldsymbol{M}'$, denoted as $\boldsymbol{M}]>\boldsymbol{M}'$. The new marking $\boldsymbol{M}'$ could be calculated as follows:

$$\boldsymbol{M}'(p)=\begin{cases}0, & p\in{}^{\bullet}t-t^{\bullet},\\ (\sum \boldsymbol{M}(p_i)\times w(p_i,t))\times \mathrm{CF}(t,p), & p\in t^{\bullet}-{}^{\bullet}t,\\ \boldsymbol{M}(p), & p\notin{}^{\bullet}t\cup t^{\bullet}.\end{cases} \tag{2}$$

Definition 5 (Incidence matrix) An incidence matrix $\boldsymbol{H}$ is used to represent the entire flow relationships between P and T of the FPN, the element of which satisfies

$$h_{ij}=\begin{cases}-1, & \text{if } p_i\in \boldsymbol{I}(t_j), p_i\in P, t_j\in T,\\ 1, & \text{if } p_i\in \boldsymbol{O}(t_j), p_i\in P, t_j\in T\\ & (i=1, 2, \cdots, n; j=1, 2, \cdots, m),\\ 0, & \text{otherwise.}\end{cases} \tag{3}$$

Definition 6 (Route of place and the longest route of FPN) For a given $n\times m$ FPN, the transition string t_1, t_2, ⋯, t_j can be fired in order. For a given place p, if p can obtain the token from the input place, the transition string t_1, t_2, ⋯, t_j is called a route of place p, and the length l_j of route l of place p is defined as $l_j=|r_j|=h_j$, where h is the number of the transitions of this route r_j.

If p is the output place, there are k routes of p from the input place r_1, r_2, ⋯, r_k, and the corresponding path length of each route is l_1, l_2, ⋯, l_k, respectively. The longest route of the FPN is $l_{\max}=|\max(l_1, l_2, \cdots, l_k)|$.

Definition 7 (Immediate reachability set and reachability set) If $p_i\in{}^{\bullet}t_i$ and $p_j\in t_i^{\bullet}$, then p_j is said immediately reachable from p_i. The set of places that are immediately reachable from a place p_i is called the immediate reachability set of p_i and is denoted by $\mathrm{IRS}(p_i)$. The set of places that are reachable from place p_i is called the reachability set of p_i and is denoted by $\mathrm{RS}(p_i)$.

2.2 Fuzzy production rules

Definition 8 (FPR) An FPR is generally defined below.

If $D(\lambda)$, then Q (CF, μ, w), where:

(1) D is a finite set of preconditions $D=\{D_1, D_2, \cdots, D_n\}$;

(2) Q is a finite set of conclusions $Q=\{Q_1, Q_2, \cdots, Q_n\}$;

(3) $\mathrm{CF}\in(0,1]$ is the belief strength of a rule to indicate the confidence of the related conclusions derived by conditions;

(4) $\mu\in(0,1]$ is the threshold value of the rule;

(5) $w\in(0,1]$ is the weight of each precondition.

An FPR is a type of production rule that describes the interior relationship between pre-positions and conclusions with fuzzy parameters. FPRs can be classified into three types: simple rules, "AND" rules, and "OR" rules.

(1) Simple rule

If $D(\lambda)$, then Q (CF, μ, w=1).

If $\lambda\geq\mu$ exists, then the rule can be fired. The corresponding FPN of a simple rule is generated as shown in Fig. 1a, and the result after being fired is $m(p_j)=m(p_i)\times w_i\times$CF (Fig. 1b).

(2) "AND" rule

If $D_1(\lambda_1)$ and $D_2(\lambda_2)$ and ⋯ and $D_n(\lambda_n)$, then Q (CF, μ, $\sum w_i=1$).

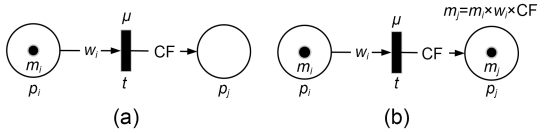

Fig. 1 Fuzzy Petri net (FPN) corresponds to the simple rule: (a) before transition fired; (b) after transition fired

If $\sum w_i\times\lambda_i\geq\mu$ exists, then the rule can be fired. The corresponding FPN of the "AND" rule is generated as shown in Fig. 2a, and the result after being fired is $m(p_j)=\sum(m(p_i)\times w_i)\times$CF (Fig. 2b).

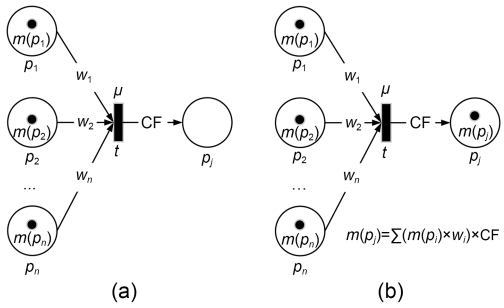

Fig. 2 FPN corresponds to the "AND" rule: (a) before being fired; (b) after being fired

(3) "OR" rule

If $D_1(\lambda_1)$ or $D_2(\lambda_2)$ or ⋯ or $D_n(\lambda_n)$, then Q (CF, μ, w_i=1).

If $w_i\times\lambda_i\geq\mu_i$ exists, then the rule can be fired. The corresponding FPN of the "OR" rule is generated as shown in Fig. 3a, and the result after being fired is $m(p_j)=\max(m(p_i)\times w_i)\times\mathrm{CF}_i$ (Fig. 3b).

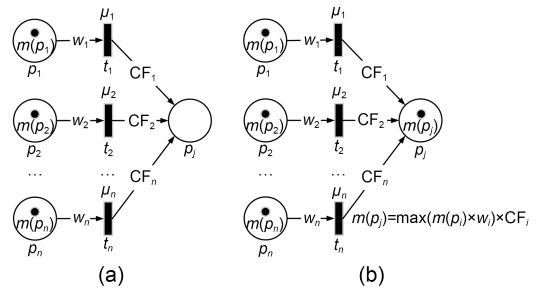

Fig. 3 FPN corresponds to the "OR" rule: (a) before being fired; (b) after being fired

2.3 Incidence matrix

An incidence matrix of a given FPN records the flow relationships between places and transitions. Specifically, the incidence matrix is used to delineate different FPRs. Consequently, the function and associated analysis of the incidence matrix are discussed below.

A specific FPN is illustrated in Fig. 4. The incidence matrix $\boldsymbol{H}$ corresponding to Fig. 4 is given in Eq. (4).

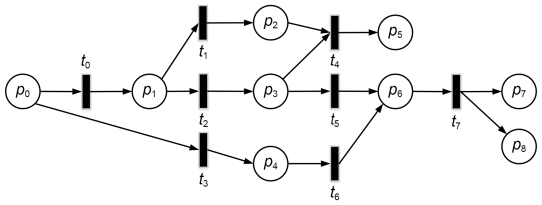

Fig. 4 FPN model

$$\boldsymbol{H}=\begin{array}{c} \\ p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{array}\begin{array}{c} \begin{array}{cccccccc} t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}. \quad (4)$$

According to the incidence matrix $\boldsymbol{H}$, the rules are interpreted as follows:

(1) If a column contains multiple elements with a value of −1, this indicates that the corresponding transition has numerous inputs in the FPN, and the FPR represents an "AND" rule. For example, the column corresponding to t_4 in the incidence matrix $\boldsymbol{H}$ exemplifies this.

(2) If a row contains multiple elements with a value of 1, it indicates that the corresponding place is the output of several different transitions in the FPN, and the corresponding FPR is an "OR" rule. For example, the row corresponding to p_6 in the incidence matrix $\boldsymbol{H}$ exemplifies this.

(3) If a column contains multiple elements with a value of 1, it indicates that the corresponding transition has multiple outputs in the FPN, which corresponds to the FPN with the disjunctive conclusion rule. For example, the column corresponding to t_7 in the incidence matrix $\boldsymbol{H}$ exemplifies this.

(4) If a row contains multiple elements with a value of −1, it indicates that the place has multiple outputs in the FPN, which corresponds to the FPN with the disjunctive conclusion rule. For example, the row corresponding to p_1 in the incidence matrix $\boldsymbol{H}$ exemplifies this.

2.4 General reasoning algorithm using FPN and related operational operators

According to the references (Gao et al., 2003; Yuan et al., 2008), six operators are frequently used in reasoning using FPN below.

$\oplus$: $\boldsymbol{C}=\boldsymbol{A}\oplus\boldsymbol{B}\Rightarrow c_{ij}=\max(a_{ij}, b_{ij})$, where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are all $n\times m$ fuzzy matrices with a_{ij}, b_{ij}, and c_{ij} being their elements, respectively;

$\otimes$: $\boldsymbol{C}=\boldsymbol{A}\otimes\boldsymbol{B}\Rightarrow c_{ij}=\max\limits_{1\le k\le l}(a_{ik}\cdot b_{kj})$, where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are $n\times l$, $l\times m$, and $n\times m$ fuzzy matrices with a_{ij}, b_{ij}, and c_{ij} being their elements, respectively;

$\odot$: $\boldsymbol{C}=\boldsymbol{A}\odot\boldsymbol{B}\Rightarrow c_{ij}=a_{ij}.b_{ij}$, where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are all $n\times m$ fuzzy matrices with a_{ij}, b_{ij}, and c_{ij} being their elements, respectively;

©: $\boldsymbol{C}=\boldsymbol{A}©\boldsymbol{B}\Rightarrow\begin{cases}c_{ij}=1, a_{ij}\ge b_{ij}\\ c_{ij}=0, a_{ij}<b_{ij}\end{cases}$, where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are all $n\times m$ fuzzy matrices with a_{ij}, b_{ij}, and c_{ij} being their elements, respectively;

∇: $\boldsymbol{C}=\boldsymbol{A}\nabla\boldsymbol{B}\Rightarrow c_{ij}=\min(a_{ij}, b_{ij})$, where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are all fuzzy matrices with a_{ij}, b_{ij}, and c_{ij} being their elements, respectively (more importantly, the elements of matrix $\boldsymbol{B}$ are all 1);

® $\boldsymbol{C}=\boldsymbol{A}\Rightarrow\begin{cases}c_{ij}=a_{ij}^{-1}, a_{ij}\ne 0\\ c_{ij}=0, a_{ij}=0\end{cases}$, where $\boldsymbol{A}$ is an $n\times m$ fuzzy matrix.

The general reasoning algorithm using FPN is illustrated in Algorithm 1.

Algorithm 1 Formal reasoning algorithm by FPN

Input: Initial fuzzy token value $\boldsymbol{M}_0$, input matrix $\boldsymbol{I}$, output matrix $\boldsymbol{O}$, and threshold vector **Th**.

Output: Final token value $\boldsymbol{M}$ of the places.

Step 1 Initialize the input variables and the number of iterations k=0.

Step 2 Calculate the equivalent fuzzy input token value vector $\boldsymbol{E}$ for each transition:

$$\boldsymbol{E}=(\boldsymbol{I}^{\mathrm{T}}\times\boldsymbol{M}_k)\nabla\boldsymbol{B}, \tag{5}$$

where $\boldsymbol{B}$ is an m-dimensional column vector with all elements of 1.

Step 3 Compare the acquired token value vector and threshold vector **Th**, and remove the input items that cannot fire transitions:

$$\boldsymbol{G}=\boldsymbol{E}\odot(\boldsymbol{E}©\mathbf{Th}). \tag{6}$$

Step 4 Calculate the token value of the fuzzy output places:

$$\boldsymbol{S}=(\boldsymbol{O}\otimes\boldsymbol{G})\nabla\boldsymbol{B}, \tag{7}$$

where $\boldsymbol{B}$ is an m-dimensional column vector with all elements of 1.

Step 5 Calculate the token value for all currently obtained places:

$$\boldsymbol{M}_{k+1}=\boldsymbol{M}_k\oplus\boldsymbol{S}. \tag{8}$$

Step 6 If $\boldsymbol{M}_{k+1}\ne\boldsymbol{M}_k$, make k++ and move to Step 2; otherwise, the reasoning ends, and output token value $\boldsymbol{M}_{k+1}$ and the number of iterations k.

3 Reversed FPN (Re-FPN) and related operations

This section introduces the related concepts of the Re-FPN and the corresponding mapping between the original FPN and Re-FPN. Definitions of reverse nets and Re-FPN are illustrated below.

Definition 9 (Reverse PN (Hu et al., 2011)) Let $\Sigma_1=(P_1, T_1, F_1)$ and $\Sigma_2=(P_2, T_2, F_2)$ be two PNs; Σ_2 is the corresponding reverse PN of Σ_1 if $F_2=\{(x, y)|(y, x)\in F_1\}$ exists.

According to Definition 9, the formalism of the Re-FPN could be given.

Definition 10 (Re-FPN) Let $\Sigma_1=(P_1, T_1, F_1, \boldsymbol{M}_1, \boldsymbol{W}_1, \mathbf{Th}_1, \mathbf{CF}_1, \boldsymbol{I}_1, \boldsymbol{O}_1)$ and $\Sigma_2=(P_2, T_2, F_2, \boldsymbol{M}_2, \boldsymbol{W}_2, \mathbf{Th}_2, \mathbf{CF}_2, \boldsymbol{I}_2, \boldsymbol{O}_2)$ be two FPNs; Σ_2 is the corresponding Re-FPN of Σ_1 if $P_1=P_2$, $T_1=T_2$, and $F_1=F_2^{-1}$ exist.

3.1 Correspondence between original FPN and Re-FPN

To derive the corresponding Re-FPN from the original FPN, some key points are considered from various perspectives, including the Re-FPN that corresponds to the simple rule, the Re-FPN that corresponds to the "AND" rule, the Re-FPN that corresponds to the "OR" rule, the Re-FPN with disjunctive conclusions, and the Re-FPN with conjunctive conclusions. Additionally, during the generation of the Re-FPN, a potential issue arises where the input and output arc weights may exceed 1, following reversal operations. To construct a Re-FPN that mirrors the forward reasoning rule, two new matrices are considered, which are the weight matrix of the input place **In** and the weight

matrix of the output place **Out**. These matrices are designed to maintain consistency within the transformed net system with the definition of the FPN.

Definition 11 (Weight matrix of the input place **In**) **In**=$(\alpha(p_i, t_j))_{n\times m}$ is used to record the weights associated with the input places. $\alpha(p_i, t_j)=w_{ij}$ when there is a directed arc from p_i to t_j (i=1, 2, ⋯, n; j=1, 2, ⋯, m); otherwise, $\alpha(p_i,t_j)$=0.

Definition 12 (Weight matrix of the output place **Out**) **Out**=$(\beta(p_i, t_j))_{n\times m}$ is used to record the CF_{ij} associated with the output places. $\beta(p_i,t_j)=CF_{ij}$ when there is a directed arc from t_j to p_i (i=1, 2, ⋯, n; j=1, 2, ⋯, m); otherwise, $\beta(p_i,t_j)$=0.

The corresponding Re-FPN generation for the five different types of FPRs is detailed as follows:

(1) Re-FPN corresponding to the simple rule

The Re-FPN corresponding to the simple rule is shown in Fig. 5a. If $m(p_j)\times(CF'\times\beta)\geq\mu$ exists, the transitions can be fired, and the result after being fired is $m(p_i)=m(p_j)\times(CF'\times\beta)\times(\alpha_i\times w_i')$ (Fig. 5b), where $\alpha_i=\frac{1}{w_i}$, w_i'=1, CF′=1, and $\beta=\frac{1}{CF}$ (i=1, 2, ⋯, n).

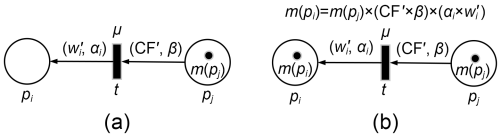

Fig. 5 Re-FPN corresponds to the simple rule: (a) before transition fired; (b) after transition fired

(2) Re-FPN corresponds to the "AND" rule

The Re-FPN corresponding to the "AND" rule is shown in Fig. 6a, where the new output weight is $w_i'=\frac{1}{n}$. The transitions can be fired if $m(p_j)\times(CF'\times\beta)\geq\mu$ exists. The result after being fired is $m(p_i)=m(p_j)\times(CF'\times\beta)\times(\alpha_i\times w_i')$ (Fig. 6b), where $\alpha_i=\frac{1}{w_i}$, $w_i'=\frac{1}{n}$, CF′=1, and $\beta=\frac{1}{CF}$ (i=1, 2, ⋯, n).

(3) Re-FPN corresponds to the "OR" rule

The Re-FPN corresponding to the "OR" rule is shown in Fig. 7a. The transitions can be fired if $m(p_j)\times(CF_i'\times\beta_i)\geq\mu_i$ exists. The result after being fired is $m(p_i)=m(p_j)\times(CF_i'\times\beta_i)\times(\alpha_i\times w_i')$ (Fig. 7b), where $\alpha_i=\frac{1}{w_i}$, w_i'=1, CF_i'=1, and $\beta_i=\frac{1}{CF_i}$ (i=1, 2, ⋯, n).

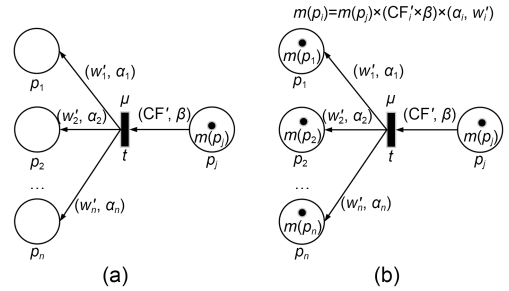

Fig. 6 Re-FPN corresponds to the "AND" rule: (a) before being fired; (b) after being fired

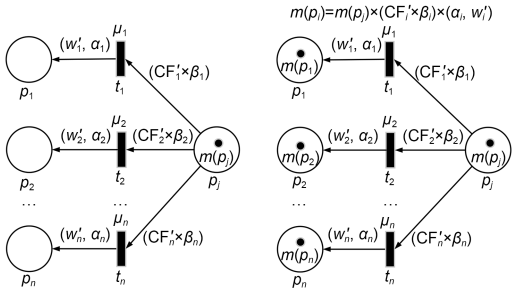

Fig. 7 Re-FPN corresponds to the "OR" rule: (a) before being fired; (b) after being fired

(4) Re-FPN corresponding to FPN with disjunctive conclusions

The FPN model with disjunctive conclusions could be understood as consisting of several simple rules with the same preconditions as shown in Fig. 8. Its result after firing in the simple rule way is $m(p_i)=m(p_j)\times w_i\times CF_i$. The corresponding Re-FPN is shown in Fig. 9a. The transitions can be fired if $m(p_i)\times(CF_i'\times\beta_i)\geq\mu_i$ exists. Moreover, if multiple transitions can be fired simultaneously, the fired result takes the maximum value of each output $m(p_j)=\max(m(p_i)\times(CF_i'\times\beta_i)\times(\alpha_i\times w_i'))$ (Fig. 9b), where $\alpha_i=\frac{1}{w_i}$, w_i'=1, CF_i'=1, and $\beta_i=\frac{1}{CF_i}$ (i=1, 2, ⋯, n).

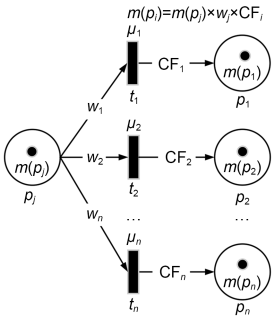

Fig. 8 FPN with disjunctive conclusion

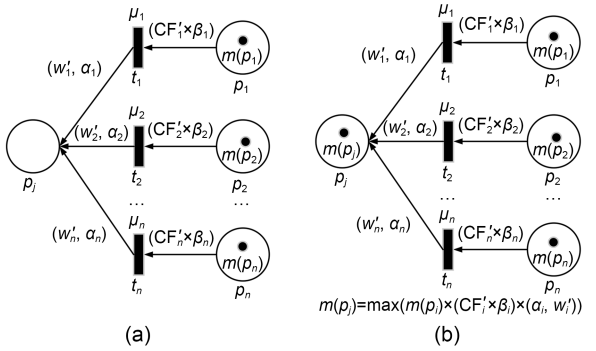

Fig. 9 Re-FPN corresponds to Fig. 8: (a) before being fired; (b) after being fired

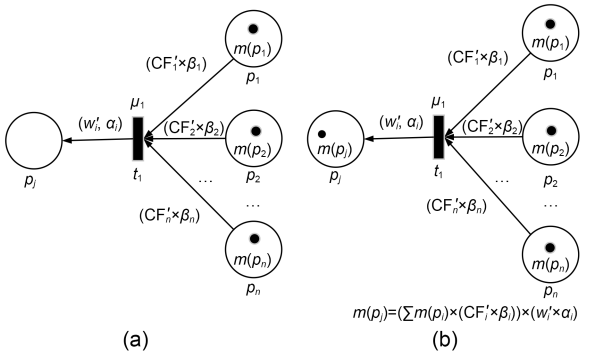

Fig. 11 Re-FPN corresponds to Fig. 10: (a) before being fired; (b) after being fired

(5) Re-FPN corresponding to the FPN with conjunctive conclusions

The FPN with conjunctive conclusions can be understood as being composed of several simple rules with the same precondition as shown in Fig. 10. Its result after firing in the simple rule way is $m(p_i)=m(p_j)\times w_i\times \mathrm{CF}_i$. The corresponding Re-FPN is shown in Fig. 11a. The transitions can be fired; the fired result is $m(p_j)=(\Sigma m(p_i)\times(\mathrm{CF}'_i\times\beta_i))\times(w'_i\times\alpha_i)$ if $\Sigma m(p_i)\times(\mathrm{CF}'_i\times\beta_i)\geq\mu_i$ exists (Fig. 11b), where $\alpha_i=\frac{1}{w_i}$, $w'_i=1$, $\mathrm{CF}'_i=\frac{1}{n}$, and $\beta_i=\frac{1}{\mathrm{CF}_i}$ (i=1, 2, ⋯, n). To satisfy the condition that the sum of the weights of the individual preconditions of the "AND" rule is 1, it needs to normalize the input weights after the reverse operation $\mathrm{CF}'_i=\frac{1}{n}$.

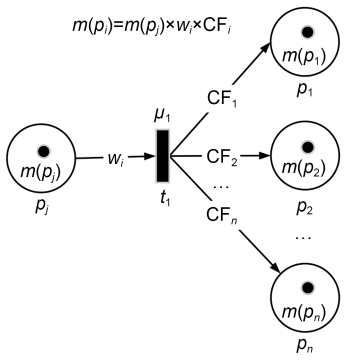

Fig. 10 FPN with the conjunctive conclusion

3.2 Re-FPN generation algorithm

Due to the mapping relationship, the Re-FPN generation algorithm is demonstrated in Algorithm 2.

Algorithm 2 Re-FPN generation algorithm

Input: FPN(Σ)=(P_1, T_1, F_1, $\boldsymbol{M}$, $\boldsymbol{W}$, $\mathbf{Th}_1$, $\mathbf{CF}_1$, $\boldsymbol{I}_1$, $\boldsymbol{O}_1$), incidence matrix $\boldsymbol{H}$.

Output: Re-FPN's places P_2, transitions T_2, flow relationship F_2, threshold vector $\mathbf{Th}_2$, confidence matrix $\mathbf{CF}_2$, input matrix $\boldsymbol{I}_2$, output matrix $\boldsymbol{O}_2$, weight matrix of the input place **In**, and weight matrix of the output place **Out**.

Step 1 $P_2=P_1$, $T_2=T_1$, $F_2=F_1^{-1}$.

Step 2 $\boldsymbol{I}_2=\boldsymbol{I}_3=\boldsymbol{H}\oplus\boldsymbol{O}_1$.

Step 3 **In**=®$\boldsymbol{O}_1$.

Step 4 $T'=\{t_i|\mathrm{count}(\boldsymbol{H}(:,i)>0)>1\}$($i$=0, 1, ⋯, $m-1$) represents the transition set with the "conjunctive conclusion" rule. Moreover, count($\boldsymbol{H}(:,i)>0$) denotes the number of elements greater than 0 in column i of the incidence matrix $\boldsymbol{H}$.

Step 5 If $\exists t_j\in T'$, num=count($\boldsymbol{H}(:,j)>0$), $I_2(i,j)=\frac{I_3(i,j)}{\mathrm{num}}$ (i=0, 1, ⋯, $n-1$); otherwise, move to Step 6.

Step 6 $\boldsymbol{O}_2=\boldsymbol{O}_3=-\boldsymbol{H}\oplus\boldsymbol{I}_1$.

Step 7 **Out** =®$\boldsymbol{I}_1$.

Step 8 $T'=\{t_i|\mathrm{count}(\boldsymbol{H}(:,i)>0)>1\}$($i$=0, 1, ⋯, $m-1$) represents the transition set with the "AND" rule. Moreover, count($\boldsymbol{H}(:,i)<0$) denotes the number of elements less than 0 in column i of the incidence matrix $\boldsymbol{H}$.

Step 9 If $\exists t_j\in T'$, num=count($\boldsymbol{H}(:,i)<0$); $O_2(i,j)=\frac{O_3(i,j)}{\mathrm{num}}$ (i=0, 1, ⋯, $n-1$); otherwise, move to Step 10.

Step 10 Threshold vector $\mathbf{Th}_2=\mathbf{Th}_1$.

Step 11 Confidence matrix $\mathbf{CF}_2=\boldsymbol{O}_2$.

Step 12 Output Re-FPN's places P_2, transitions T_2, flow relationship F_2, input matrix $\boldsymbol{I}_2$, output matrix $\boldsymbol{O}_2$, threshold vector $\mathbf{Th}_2$, confidence matrix $\mathbf{CF}_2$, weight matrix of the input place **In**, and weight matrix of the output place **Out**.

In general, assuming that the FPN has n places and m transitions, the time complexity of Algorithm 2 is $O(nm)$.

4 Parallel bidirectional reasoning algorithm

The main stages of the proposed PBR-HFPN-RDD algorithm are outlined as follows:

(1) Obtain the hierarchical FPN (HFPN) for subsequent decomposition operations;

(2) Decompose the HFPN into two sub-FPNs (left-sub-FPN and right-sub-FPN);

(3) Generate the Re-right-sub-FPN for the right-sub-FPN;

(4) Execute parallel reasoning between the left-sub-FPN and the Re-right-sub-FPN.

4.1 Hierarchical FPN

In some cases, the affiliation relationship between places and transitions in FPNs is not clear. This could lead to confusion regarding the hierarchical structure of the FPN and make it difficult to determine the sub-FPNs to which the places and transitions belong when decomposing the original FPN. To address this issue, this study employs the hierarchical algorithm by reverse search (HFPN-RS) to obtain the HFPN (Xiang et al., 2023).

4.2 Dynamic decomposition of HFPN

For the FPN depicted in Fig. 12, assume that the initial marking is $\boldsymbol{M}_0=(0.4, 0.6, 0)^{\mathrm{T}}$. The result of enabling and firing based on the "AND" rule is $\boldsymbol{M}_1=(0.4, 0.6, 0.437)^{\mathrm{T}}$. Conversely, the initial marking of the Re-FPN corresponding to the "AND" rule is $\boldsymbol{M}_0=(0, 0, 0.437)^{\mathrm{T}}$, and the result after enabling and firing according to the reverse rule is $\boldsymbol{M}_1=(0.3286, 0.7667, 0.437)^{\mathrm{T}}$. Clearly, the token values of places p_0 and p_1 obtained through reasoning in the Re-FPN do not align with the original FPN.

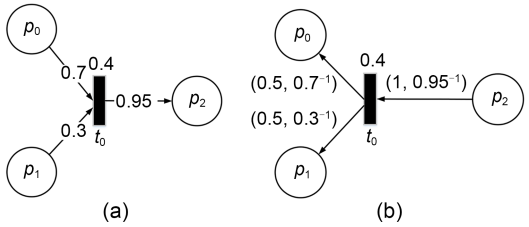

Fig. 12 Example of FPN: (a) FPN corresponds to the "AND" rule; (b) Re-FPN corresponds to Fig. 12a

Due to the inability to determine the exact weights of the FPN corresponding to the "AND" rule after reversal, it is impractical to obtain fuzzy token values for the Re-FPN that are consistent with the original FPN. This inconsistency becomes the primary source of error in the reasoning process of the Re-FPN. Consequently, we propose a dynamic decomposition strategy for the HFPN to mitigate the errors in the reasoning process of the Re-FPN. For instance, Fig. 13 illustrates three decomposition strategies for the HFPN. After executing the HFPN-RS, the sets of transitions for each layer are defined as $T_1=\{t_0, t_1, t_2\}$, $T_2=\{t_3, t_4, t_5\}$, $T_3=\{t_6, t_7\}$, and $T_4=\{t_8\}$.

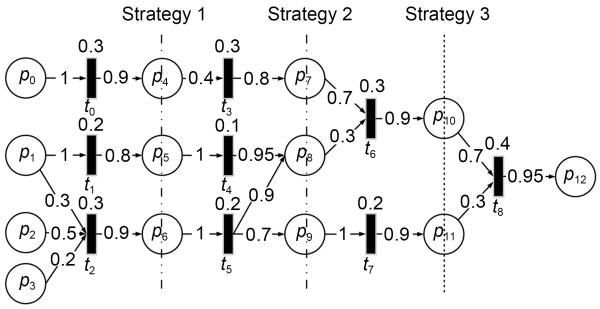

Fig. 13 Different decomposition strategies

Based on the method of distinguishing between different rules, if there are multiple inputs for t_2 within the set $T_1=\{t_0, t_1, t_2\}$, this indicates that the rule corresponds to the "AND" rule. Similarly, the number of "AND" rules within the set T_1 is 1, the number of "AND" rules within the set T_2 is 0, the number of "AND" rules within the set T_3 is 1, and the number of "AND" rules within the set T_4 is 1.

If the subsequent set T_1 is used as the basis for decomposition, as in decomposition strategy 1 shown in Fig. 13, it is divided into a left-sub-FPN and a right-sub-FPN. In the right-sub-FPN, as depicted in Fig. 14, there are two "AND" rules.

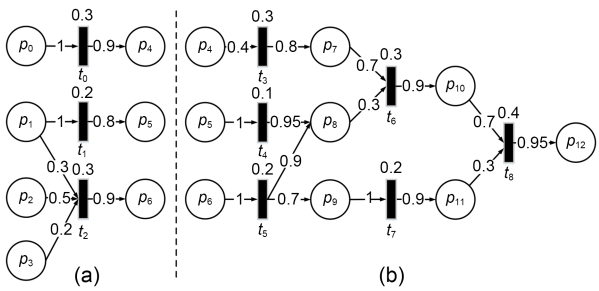

Fig. 14 Hierarchical FPN (HFPN) model decomposition schematic diagram: (a) left-sub-FPN; (b) right-sub-FPN

If the subsequent set T_2 is used as the basis for decomposition, as in decomposition strategy 2 shown in Fig. 13, it is divided into a left-sub-FPN and a right-sub-FPN, with the right-sub-FPN containing two "AND" rules.

If the subsequent set T_3 is used as the basis for decomposition, as in decomposition strategy 3 shown in Fig. 13, it is divided into a left-sub-FPN and a right-sub-FPN, with the right-sub-FPN having only one "AND" rule.

In summary, dynamically decomposing the FPN according to practical needs, such as the size of the FPN or the types of rules in the sub-FPNs, can reduce errors in the reasoning process. Based on the above analysis, the proposed dynamic decomposition algorithm is given in Algorithm 3.

Algorithm 3: Dynamic decomposition algorithm of HFPN

Input: HFPN input matrix $\boldsymbol{I}$, output matrix $\boldsymbol{O}$, incidence matrix $\boldsymbol{H}$, the set of variants for each layer $T_k=\{T_1, T_2, \cdots, T_k\}$.

Output: left-sub-FPN, right-sub-FPN, the input matrices $\boldsymbol{I}_1$, $\boldsymbol{I}_2$, the output matrices $\boldsymbol{O}_1$, $\boldsymbol{O}_2$, and the threshold vectors $\mathbf{Th}_1$, $\mathbf{Th}_2$.

Step 1 Initialize parameters $A[k]=\{0\}$, $C[k]=\{0\}$.

Step 2 If $k=1$, then the FPN does not need to be decomposed and exits.

Step 3 Count the number of "AND" rules in each level of the transition set A: If $t_j \in T_i \cup (\text{count}(\boldsymbol{H}(:, j)<0)>1)$ ($j=0, 1, \cdots, m-1$; $i=1, \cdots, k$), then $A[i]=A[i]+1$.

Step 4 Count the number of "AND" rules in the left-sub-FPN under different decomposition strategies:

$$C[i]=\sum_{j=1}^{i} A[j] \ (i=1, 2, \cdots, k). \tag{9}$$

Step 5 The decomposition strategy is determined according to the principles of the minimum number of "AND" rules and the maximum number of layers in the right-sub-FPN ($R_r \le \frac{k}{2}$, and R_r denotes the number of layers of the right-sub-FPN);

$$\text{minIndex}=\text{argmin}\left(C[k]-C[i]\right)\left(i=\left\lfloor \frac{k}{2} \right\rfloor, \cdots, k-1\right), \tag{10}$$

where argmin( ) represents returning the index corresponding to the variable that achieves the minimum value.

Step 6 The elements of the left-sub-FPN are composed of layers T_1–T_{minIndex} and the pre-set and post-set places of each layer, and the elements of the right-sub-FPN are composed of layers T_{minIndex}–T_k and the pre-set and post-set places of each layer.

Assuming that the HFPN has n places and m transitions, the time complexity of Algorithm 3 is $O(nm)$.

4.3 Proposed PBR-HFPN-RDD algorithm

The detailed steps of the PBR-HFPN-RDD algorithm are shown in Algorithm 4.

Algorithm 4 PBR-HFPN-RDD algorithm

Input: Initial marking $\boldsymbol{M}_0$, input matrix $\boldsymbol{I}$, output matrix $\boldsymbol{O}$, and threshold vector **Th**.

Output: The fuzzy token value of the predicted initial places ρ_0 and the error value E.

Pre-processing: If there exists a loop structure of FPN, the PBR-HFPN-RDD will be exited automatically.

Step 1 Initialize the input variables, and make the number of iterations $i=0$.

Step 2 Judge whether the hierarchical algorithm is needed for a given FPN and implement the hierarchical operations. Then, update the input matrix $\boldsymbol{I}'$, output matrix $\boldsymbol{O}'$, and $\mathbf{Th}'$.

Step 3 Use Algorithm 3 to obtain the left-sub-FPN and right-sub-FPN, the input matrices $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, the output matrices $\boldsymbol{O}_1$ and $\boldsymbol{O}_2$, and the threshold vectors $\mathbf{Th}_1$ and $\mathbf{Th}_2$.

Step 4 Generate the Re-right-sub-FPN corresponding to the right-sub-FPN by Algorithm 2 and obtain the related $\boldsymbol{I}'_2$, $\boldsymbol{O}'_2$, $\mathbf{Th}'_2$, **In**, and **Out**.

Step 5 Implement Algorithm 1 on the left-sub-FPN to obtain the inference result θ_1 and final value of the output place(s) of the left-sub-FPN V_1.

Step 6 Obtain the equivalent weighted input and output place matrices of the Re-right-sub-FPN, which are $\boldsymbol{I}'_2=\boldsymbol{I}'_2 \odot \mathbf{In}$ and $\boldsymbol{O}'_2=\boldsymbol{O}'_2 \odot \mathbf{Out}$, respectively. The input place set P_{in} of the Re-right-sub-FPN is also obtained, where $P_{\text{in}}=\{p_0, p_1, \cdots, p_{y-1}\}$ and the number of input places is y.

Step 7 Initialize the initial token value of the Re-right-sub-FPN $\boldsymbol{\rho}_i^{(0)}=(0, 0, \cdots, 0)^{\mathrm{T}}$.

Step 8 Make the initial token value of the input place p_i equal to 1 and implement Algorithm 1 on the Re-right-sub-FPN to obtain the inference result in $\theta_r^{(i)}$ and the final value of the output place(s) of the right-sub-FPN $V_r^{(i)}$.

Step 9 Calculate the error in the output places between the left-sub-FPN and Re-right-sub-FPN as $E_i=\sqrt{\sum_{j=1}^{k}(V_{r(j)}^{(i)}-V_{1(j)})^2}$, where k is the number of the output places of sub-FPN.

Step 10 If $i \ne y-1$, make $i=i+1$ and move to Step 7; otherwise, output $E_x=\min(E_0, E_1, \cdots, E_{y-1})$ and $\boldsymbol{\rho}_x^{(0)}$. The corresponding input place which fulfills $m(p_x)=1$ is the final result of the whole reasoning.

4.4 Algorithm analysis

Assuming that there is an FPN without a loop with n places and m transitions, the reasoning executes $h+1$

times under the worst conditions, where h is the number of transitions of the longest path in the FPN.

At the identical time $h \leqslant m$, the time complexity is: $O(nm^2)$ for formal reasoning using FPN; $O(m(n+m))$ for the hierarchical algorithm, and the size of the FPN increases from $n \times m$ to $(n+r) \times (m+r)$ after performing the hierarchical operation, where r is the number of virtual-place–virtual-transition pairs to be added; $O(nm)$ for the HFPN decomposition algorithm; $O(nm)$ for the generation of the Re-right-sub-FPN; $\max[O(gl^2), O(sx(n+r-g)(m+r-l))]$ for executing bidirectional reasoning, where g, l, s, x are the left-sub-FPN place number, left-sub-FPN transition number, right-sub-FPN reasoning loop times, and right-sub-FPN execution formal reasoning algorithm cycle times, respectively.

Hence, the entire time complexity of the proposed PBR-HFPN-RDD algorithm is

$$\begin{aligned} &O\{m(n+m)+nm+nm+\\ &\max[O(gl^2), O(sx(n+r-g)\\ &\cdot(m+r-l))]\} \leqslant O\{m(n+m)+nm+nm+\\ &\max[O(gl^2), O(s(m+r)(n+r-g)\\ &\cdot(m+r-l))]\} \leqslant O(s(n+r)(m+r)^2). \end{aligned}$$

To summarize, for a given FPN with n places and m transitions, in the worst case, the time complexity of PBR-HFPN-RDD is $O(s(n+r)(m+r)^2)$, where s is the number of output places and r is the number of virtual-place–virtual-transition pairs to be added.

5 Experiment and analysis

Fault diagnosis plays a crucial role in industrial manufacturing equipment and production processes. It is a critical maintenance activity that aims to detect and identify problems in equipment or production lines in a timely manner to reduce downtime, increase production efficiency, and ensure product quality. In this section, an experimental case of fault diagnosis is used to verify the correctness of the proposed algorithm.

The assumed rules from the fault diagnosis rule base are listed below.

R_0: IF d_0 THEN d_5 (μ=0.3, CF=0.9, w=1).

R_1: IF d_1 THEN d_6 (μ=0.2, CF=0.8, w=1).

R_2: IF d_1 AND d_2 AND d_3 THEN d_7 (μ=0.3, CF=0.9, w_1=0.3, w_2=0.5, w_3=0.2).

R_3: IF d_4 AND d_5 THEN d_8 (μ=0.3, CF=0.8, w_1=0.6, w_2=0.4).

R_4: IF d_6 THEN d_9 (μ=0.1, CF=0.95, w=1).

R_5: IF d_7 THEN d_9 AND d_{10} (μ=0.2, CF_1=0.9, CF_2=0.7, w=1).

R_6: IF d_8 AND d_9 THEN d_{11} (μ=0.3, CF=0.9, w_1=0.7, w_2=0.3).

R_7: IF d_{10} THEN d_{12} AND d_{13} (μ=0.2, CF_1=0.9, CF_2=0.8, w=1).

R_8: IF d_{11} AND d_{12} THEN d_{14} (μ=0.4, CF=0.95, w_1=0.7, w_2=0.3).

R_9: IF d_{13} THEN d_{15} (μ=0.3, CF=0.8, w=1).

The corresponding FPN model is generated as shown in Fig. 15.

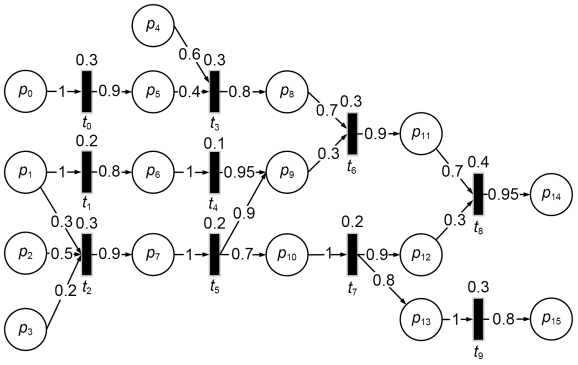

Fig. 15 Corresponding FPN of the given rules

5.1 FPN hierarchical phase

The input matrix $\boldsymbol{I}$, output matrix $\boldsymbol{O}$, and threshold vector **Th** of Fig. 15 are given in Eqs. (11)–(13).

$$\boldsymbol{I}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (11)$$

$$\boldsymbol{O}=\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0.9&0&0&0&0&0&0&0&0&0\\
0&0.8&0&0&0&0&0&0&0&0\\
0&0&0.9&0&0&0&0&0&0&0\\
0&0&0&0.8&0&0&0&0&0&0\\
0&0&0&0&0.95&0.9&0&0&0&0\\
0&0&0&0&0&0.7&0&0&0&0\\
0&0&0&0&0&0&0.9&0&0&0\\
0&0&0&0&0&0&0&0.9&0&0\\
0&0&0&0&0&0&0&0.8&0&0\\
0&0&0&0&0&0&0&0&0.95&0\\
0&0&0&0&0&0&0&0&0&0.8
\end{pmatrix}. \quad (12)$$

$$\mathbf{Th}=(0.3, 0.2, 0.3, 0.3, 0.1, 0.2, 0.3, 0.2, 0.4, 0.3)^{\mathrm{T}}. \quad (13)$$

The FPN is translated into HFPN with a clear structure by executing a hierarchical algorithm to increase the virtual place p_{16} and the virtual transitions t_{10} with three decomposition strategies, as shown in Fig. 16. The threshold of virtual transitions is 0, and the virtual transitions to the post-set place have a confidence degree of 1, i.e., CF(t, $t^{\bullet}$) ≡1.

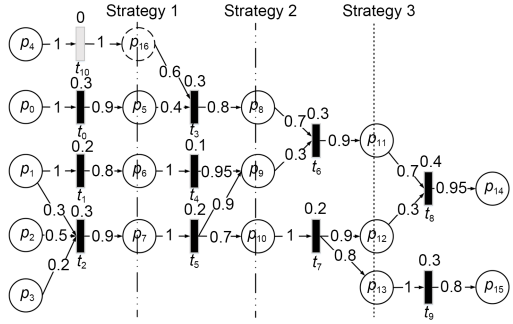

Fig. 16 FPN with hierarchical structure

The input matrix $\boldsymbol{I}'$, output matrix $\boldsymbol{O}'$, and threshold vector $\mathbf{Th}'$ of the HFPN are given by Eqs. (14)–(16).

$$\boldsymbol{I}'=\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0\\
0&1&0.3&0&0&0&0&0&0&0&0\\
0&0&0.5&0&0&0&0&0&0&0&0\\
0&0&0.2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0.4&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0.7&0&0&0&0\\
0&0&0&0&0&0&0.3&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0.7&0&0\\
0&0&0&0&0&0&0&0&0.3&0&0\\
0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0.6&0&0&0&0&0&0&0
\end{pmatrix}. \quad (14)$$

$$\boldsymbol{O}'=\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0.9&0&0&0&0&0&0&0&0&0&0\\
0&0.8&0&0&0&0&0&0&0&0&0\\
0&0&0.9&0&0&0&0&0&0&0&0\\
0&0&0&0.8&0&0&0&0&0&0&0\\
0&0&0&0&0.95&0.9&0&0&0&0&0\\
0&0&0&0&0&0.7&0&0&0&0&0\\
0&0&0&0&0&0&0.9&0&0&0&0\\
0&0&0&0&0&0&0&0.9&0&0&0\\
0&0&0&0&0&0&0&0.8&0&0&0\\
0&0&0&0&0&0&0&0&0.95&0&0\\
0&0&0&0&0&0&0&0&0&0.8&0\\
0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}. \quad (15)$$

$$\mathbf{Th}'=(0, 0.3, 0.2, 0.3, 0.3, 0.1, 0.2, 0.3, 0.2, 0.4, 0.3)^{\mathrm{T}}. \quad (16)$$

5.2 HFPN decomposition phase and Re-FPN generation phase

To reduce the error in the reasoning, the right-sub-FPN with fewer "AND" rules is obtained using the decomposition strategy 2, and the decomposition results are shown in Figs. 17a and 17b. The input

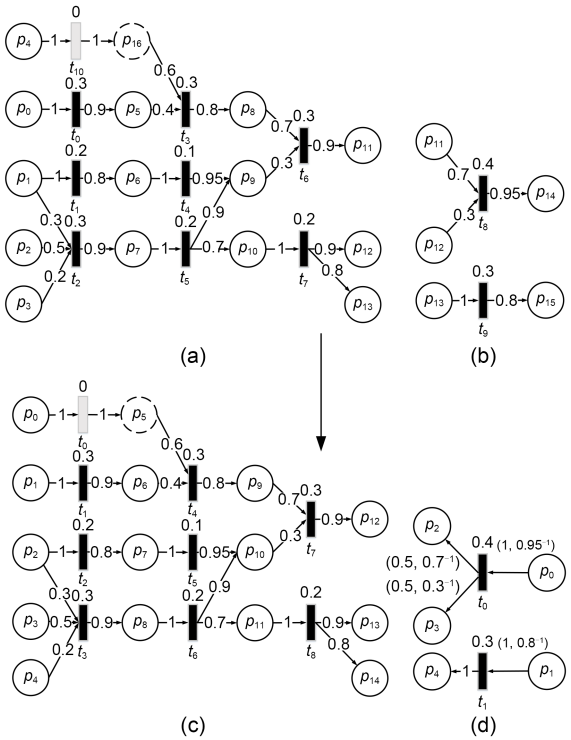

Fig. 17 HFPN decomposition process: (a) left-sub-FPN; (b) right-sub-FPN; (c) left-sub-FPN; (d) Re-right-sub-FPN

matrix $\boldsymbol{I}_1$, output matrix $\boldsymbol{O}_1$, and threshold vector $\mathbf{Th}_1$ of the left-sub-FPN are given by Eqs. (17)–(19).

$$\boldsymbol{I}_1=\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0.3&0&0&0&0&0\\0&0&0&0.5&0&0&0&0&0\\0&0&0&0.2&0&0&0&0&0\\0&0&0&0&0.6&0&0&0&0\\0&0&0&0&0.4&0&0&0&0\\0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0.7&0\\0&0&0&0&0&0&0&0.3&0\\0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{pmatrix}. \quad (17)$$

$$\boldsymbol{O}_1=\begin{pmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0\\0&0.9&0&0&0&0&0&0&0\\0&0&0.8&0&0&0&0&0&0\\0&0&0&0.9&0&0&0&0&0\\0&0&0&0&0.8&0&0&0&0\\0&0&0&0&0&0.95&0.9&0&0\\0&0&0&0&0&0&0.7&0&0\\0&0&0&0&0&0&0&0.9&0\\0&0&0&0&0&0&0&0&0.9\\0&0&0&0&0&0&0&0&0.8\end{pmatrix}. \quad (18)$$

$$\mathbf{Th}_1=(0, 0.3, 0.2, 0.3, 0.3, 0.1, 0.2, 0.3, 0.2)^{\mathrm{T}}. \quad (19)$$

Then, the right-sub-FPN will be transformed into a corresponding Re-right-sub-FPN by executing Algorithm 2, and the results are shown in Fig. 17d. The input matrix $\boldsymbol{I}'_2$, output matrix $\boldsymbol{O}'_2$, given by Eqs. (17)–(19), input places weight matrix **In**, and output place weight matrix **Out** of the right-sub-FPN are obtained as follows:

$$\boldsymbol{I}'_2=\begin{pmatrix}1&0\\0&1\\0&0\\0&0\\0&0\end{pmatrix},\quad \boldsymbol{O}'_2=\begin{pmatrix}0&0\\0&0\\0.5&0\\0.5&0\\0&1\end{pmatrix},$$

$$\mathbf{In}=\begin{pmatrix}0.95^{-1}&0\\0&0.8^{-1}\\0&0\\0&0\\0&0\end{pmatrix},\quad \mathbf{Out}=\begin{pmatrix}0&0\\0&0\\0.7^{-1}&0\\0.3^{-1}&0\\0&1\end{pmatrix}. \quad (20)$$

5.3 Parallel reasoning phase

Assume that the starting vector $\boldsymbol{M}_0$ for the given left-sub-FPN is $\boldsymbol{M}_0$=(0.5, 0.85, 0.7, 0.75, 0.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$^{\mathrm{T}}$. The final result of running the reasoning algorithm is $\boldsymbol{M}$=(0.5, 0.85, 0.7, 0.75, 0.7, 0.5, 0.765, 0.56, 0.6525, 0.4848, 0.587 25, 0.456 75, 0.463 982, 0.411 075, 0.3654)$^{\mathrm{T}}$; the final result of executing the formal reasoning algorithm is (0.463 982, 0.411 075, 0.3654)$^{\mathrm{T}}$.

Assume that the initial token value of the Re-right-sub-FPN is $\boldsymbol{\rho}^{(0)}$=(1, 0, 0, 0, 0)$^{\mathrm{T}}$, where p_0 has the maximum probability of fault. The final result of running the reasoning algorithm on the Re-right-sub-FPN could be gained as $\boldsymbol{\rho}$=(1, 0, 0.793 651, 1, 0)$^{\mathrm{T}}$.

The final token value of the output places of Re-right-sub-FPN is (0.793 651, 1, 0)$^{\mathrm{T}}$. The final error value of the token value is

$$E_1=\sqrt{\sum_{j=1}^{k}(V_{\mathrm{r}(j)}^{(i)}-V_{1(j)})^2}=0.737\,070.$$

Next, assume that the initial token value of the Re-right-sub-FPN is $\boldsymbol{\rho}^{(0)}$=(0, 1, 0, 0, 0)$^{\mathrm{T}}$, where p_1 has the maximum probability of fault. The final result of running the reasoning algorithm on the Re-right-sub-FPN could be gained as $\boldsymbol{\rho}$=(0, 1, 0, 0, 1)$^{\mathrm{T}}$. The final token value of the output places of the Re-right-sub-FPN is (0, 0, 1)$^{\mathrm{T}}$.

The final error value of the token value is

$$E_2=\sqrt{\sum_{j=1}^{k}(V_{\mathrm{r}(j)}^{(i)}-V_{1(j)})^2}=0.887\,118.$$

Because of the error value $E_1<E_2$, it further indicates that the initial place p_0 has a greater probability. The reasoning is terminated, and the result is obtained.

Since the time complexity of the formal reasoning algorithm and the bidirectional reasoning algorithm are both related to the matrix dimension, the larger the matrix dimension, the greater the time complexity. Table 1 lists a simple comparison among the scale of matrices of various FPNs.

Table 1 Matrix dimension comparison

Type of FPN	Related matrices	Scale of matrix
Original FPN	$\boldsymbol{I}$ AND $\boldsymbol{O}$	16×10
HFPN	$\boldsymbol{I}'$ AND $\boldsymbol{O}'$	17×11
left-sub-FPN	$\boldsymbol{I}_1$ AND $\boldsymbol{O}_1$	15×9
Re-right-sub-FPN	$\boldsymbol{I}'_2$ AND $\boldsymbol{O}'_2$	5×2

In Table 1, it is easy to find that the scale of the operation matrix is compressed; the scale of the original FPN matrix is 16×10, the scale of the left-sub-FPN matrix is 15×9, and the scale of the Re-right-sub-FPN matrix is 5×2 after executing the PBR-HFPN-RDD algorithm. Therefore, when performing bidirectional parallel reasoning, the matrix sizes of both the left-sub-FPN and the Re-right-sub-FPN are smaller than the size of the original FPN matrix, which effectively compresses the dimensionality of the operation matrix, reduces the time complexity of the algorithm, and improves the reasoning efficiency.

6 Conclusions and future work

In this paper, a bidirectional parallel reasoning algorithm for FPN with a focus on time complexity is proposed. Initially, the correspondence between each element of the Re-FPN and the original FPN is explored from the perspective of the fuzzy production rules. The enabling and firing rules specific to the Re-FPN are discussed in depth, and an algorithm for generating the Re-FPN (Algorithm 2) is introduced. Subsequently, a dynamic decomposition algorithm (Algorithm 3) is employed to mitigate the presence of excessive "AND" operations in the right-sub-FPN and decrease the reasoning error within the FPN. Finally, a PBR-HFPN-RDD algorithm (Algorithm 4) is developed based on the aforementioned algorithms to facilitate a bidirectional parallel inference mechanism, addressing the challenge of state explosion. The case study demonstrates that the proposed algorithm significantly solves the state explosion problem and enhances reasoning efficiency by diminishing the size of the FPN. The reasoning algorithm presented in this paper offers a novel approach to enhancing the decision-making efficiency of large-scale fault diagnosis systems.

While the proposed PBR-HFPN-RDD algorithm provides a genuine parallel inference method using the FPN to resolve the state explosion issue and yields favorable outcomes, several aspects deserve future investigation. First, in FPNs, the "AND" rule necessitates the concurrent satisfaction of all input transitions to trigger an output transition. Variations in weights signify different levels of influence of input transitions on output transition triggering. Due to this attribute, a discrepancy may arise between the token values of places activated by transitions in the reversed net and those in the original net during reasoning. Consequently, our ongoing research aims to employ techniques such as soft computing to refine parameters, including the weights of directed arcs, thereby minimizing reasoning errors. Second, the development of more adaptable, potent, and user-friendly tools to illustrate bidirectional reasoning processes is necessary. Third, the proposed bidirectional reasoning algorithm should be applied to other types of high-level network systems to broaden its application scope. Finally, the efficacy and validity of the proposed algorithm should be further tested within more complex rule-based systems.

Contributors

Yinhong XIANG designed the algorithm, drafted the paper, and finalized the paper. Diwen KANG helped organize the paper. Arezoo SARKHEYLI-HÄGELE, Yusliza YUSOFF, and Azlan Mohd ZAIN revised the paper. Kaiqing ZHOU conceptualized the main idea and led the research. All the authors had in-depth discussions.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Bai KY, Jia D, Meng WY, et al., 2023. Q-rung orthopair fuzzy Petri nets for knowledge representation and reasoning. *IEEE Access*, 11:93560-93573. https://doi.org/10.1109/ACCESS.2023.3309663

Chen RH, Yang B, Li S, et al., 2020. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Comput Ind Eng*, 149:106778. https://doi.org/10.1016/j.cie.2020.106778

Chen SM, 2000. Fuzzy backward reasoning using fuzzy Petri nets. *IEEE Trans Syst Man Cybern Part B (Cybern)*, 30(6): 846-856. https://doi.org/10.1109/3477.891146

Chen SM, 2002. Weighted fuzzy reasoning using weighted fuzzy Petri nets. *IEEE Trans Knowl Data Eng*, 14(2):386-397. https://doi.org/10.1109/69.991723

Chen XH, Zhang BK, Gao D, 2021. Bearing fault diagnosis base on multi-scale CNN and LSTM model. *J Intell Manuf*, 32(4): 971-987. https://doi.org/10.1007/s10845-020-01600-2

Gao MM, Zhou MC, Huang XG, et al., 2003. Fuzzy reasoning Petri nets. *IEEE Trans Syst Man Cybern Part A Syst Hum*, 33(3):314-324. https://doi.org/10.1109/tsmca.2002.804362

Grobelna I, Karatkevich A, 2021. Challenges in application of Petri nets in manufacturing systems. *Electronics*, 10(18): 2305. https://doi.org/10.3390/electronics10182305

Hu HS, Li ZW, Al-Ahmari A, 2011. Reversed fuzzy Petri nets and their application for fault diagnosis. *Comput Ind Eng*, 60(4):505-510. https://doi.org/10.1016/j.cie.2010.12.003

Jiang W, Zhou KQ, Sarkheyli-Hägele A, et al., 2022. Modeling, reasoning, and application of fuzzy Petri net model: a survey. *Artif Intell Rev*, 55(8):6567-6605. https://doi.org/10.1007/s10462-022-10161-0

Lakos C, Petrucci L, 2011. Modular state spaces for prioritised Petri nets. In: Calinescu R, Jackson E (Eds.), Foundations of Computer Software. Springer, Berlin, Heidelberg, p.136-156. https://doi.org/10.1007/978-3-642-21292-5_8

Lei YG, Yang B, Jiang XW, et al., 2020. Applications of machine learning to machine fault diagnosis: a review and roadmap. *Mech Syst Signal Process*, 138:106587. https://doi.org/10.1016/j.ymssp.2019.106587

Liu HC, Lin QL, Ren ML, 2013. Fault diagnosis and cause analysis using fuzzy evidential reasoning approach and dynamic adaptive fuzzy Petri nets. *Comput Ind Eng*, 66(4): 899-908. https://doi.org/10.1016/j.cie.2013.09.004

Liu HC, You JX, Li ZW, et al., 2017. Fuzzy Petri nets for knowledge representation and reasoning: a literature review. *Eng Appl Artif Intell*, 60:45-56. https://doi.org/10.1016/j.engappai.2017.01.012

Liu HC, Luan X, Zhou MC, et al., 2022. A new linguistic Petri net for complex knowledge representation and reasoning. *IEEE Trans Knowl Data Eng*, 34(3):1011-1020. https://doi.org/10.1109/TKDE.2020.2997175

Mou X, Mao LX, Liu HC, et al., 2022. Spherical linguistic Petri nets for knowledge representation and reasoning under large group environment. *IEEE Trans Artif Intell*, 3(3): 402-413. https://doi.org/10.1109/TAI.2022.3140282

Nishi T, Matsumoto I, 2015. Petri net decomposition approach to deadlock-free and non-cyclic scheduling of dual-armed cluster tools. *IEEE Trans Autom Sci Eng*, 12(1):281-294. https://doi.org/10.1109/TASE.2013.2292572

Rudin C, 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat Mach Intell*, 1(5):206-215. https://doi.org/10.1038/s42256-019-0048-x

Russell SJ, Norvig P, 2009. Artificial Intelligence: a Modern Approach (3rd Ed). Prentice Hall Press, Upper Saddle River, USA.

Salum L, 2015. Avoiding state explosion in a class of Petri nets. *Expert Syst Appl*, 42(1):519-526. https://doi.org/10.1016/j.eswa.2014.07.037

Seatzu C, 2019. Modeling, analysis, and control of automated manufacturing systems using Petri nets. Proc 24th IEEE Int Conf on Emerging Technologies and Factory Automation, p.27-30. https://doi.org/10.1109/ETFA.2019.8869012

Shen VRL, Chung YF, Chen SM, et al., 2013. A novel reduction approach for Petri net systems based on matching theory. *Expert Syst Appl*, 40(11):4562-4576. https://doi.org/10.1016/j.eswa.2013.01.057

Shi H, Liu HC, Wang JH, et al., 2022. New linguistic Z-number Petri nets for knowledge acquisition and representation under large group environment. *Int J Fuzzy Syst*, 24(8):3483-3500. https://doi.org/10.1007/s40815-022-01341-9

Shi H, Yu YX, Liu R, et al., 2024. Probabilistic linguistic Petri nets for knowledge representation and acquisition with dynamic consensus reaching process. *IEEE Trans Fuzzy Syst*, 32(4):2198-2210. https://doi.org/10.1109/TFUZZ.2023.3347436

Valmari A, 1998. The state explosion problem. In: Reisig W, Rozenberg G (Eds.), Lectures on Petri Nets I: Basic Models. Springer, Berlin, Heidelberg, p.429-528. https://doi.org/10.1007/3-540-65306-6_21

Wang XL, Lu FM, Zhou MC, et al., 2022. A synergy-effect-incorporated fuzzy Petri net modeling paradigm with application in risk assessment. *Expert Syst Appl*, 199:117037. https://doi.org/10.1016/j.eswa.2022.117037

Xiang YH, Zhou KQ, Yang SY, et al., 2023. Hierarchical algorithm of fuzzy Petri net by reverse search. *J Comput Appl*, 43(12):3676-3682 (in Chinese). https://doi.org/10.11772/j.issn.1001-9081.2022121851

Yang X, 2023. Performance analysis of Petri net based on moment generating function. *J Intell Fuzzy Syst*, 45(1):1131-1139. https://doi.org/10.3233/JIFS-231137

Ye SQ, Zhou KQ, Zain AM, et al., 2023. A modified harmony search algorithm and its applications in weighted fuzzy production rule extraction. *Front Inform Technol Electron Eng*, 24(11):1574-1590. https://doi.org/10.1631/FITEE.2200334

Yeung DS, Ysang ECC, 1998. A multilevel weighted fuzzy reasoning algorithm for expert systems. *IEEE Trans Syst Man Cybern Part A Syst Hum*, 28(2):149-158. https://doi.org/10.1109/3468.661144

Yu YX, Gong HP, Liu HC, et al., 2023. Knowledge representation and reasoning using fuzzy Petri nets: a literature review and bibliometric analysis. *Artif Intell Rev*, 56(7):6241-6265. https://doi.org/10.1007/s10462-022-10312-3

Yuan J, Shi HB, Liu C, et al., 2008. Improved basic inference models of fuzzy Petri nets. Proc 7th World Congress on Intelligent Control and Automation, p.1488-1493. https://doi.org/10.1109/WCICA.2008.4593140

Zaitsev DA, 2004. Decomposition of Petri nets. *Cybern Syst Anal*, 40(5):739-746. https://doi.org/10.1007/s10559-005-0012-0

Zeng RF, Jiang YX, Lin C, et al., 2012. Dependability analysis of control center networks in smart grid using stochastic petri nets. *IEEE Trans Parall Distrib Syst*, 23(9):1721-1730. https://doi.org/10.1109/TPDS.2012.68

Zhou KQ, Zain AM, 2016. Fuzzy Petri nets and industrial applications: a review. *Artif Intell Rev*, 45(4):405-446. https://doi.org/10.1007/s10462-015-9451-9

Zhou KQ, Zain AM, Mo LP, 2015a. A decomposition algorithm of fuzzy Petri net using an index function and incidence matrix. *Expert Syst Appl*, 42(8):3980-3990.

https://doi.org/10.1016/j.eswa.2014.12.048

Zhou KQ, Zain AM, Mo LP, 2015b. Dynamic properties of fuzzy Petri net model and related analysis. *J Cent South Univ*, 22(12):4717-4723. https://doi.org/10.1007/s11771-015-3023-7

Zhou KQ, Gui WH, Mo LP, et al., 2018. A bidirectional diagnosis algorithm of fuzzy Petri net using inner–reasoning–path. *Symmetry*, 10(6):192. https://doi.org/10.3390/sym10060192

Zhou KQ, Mo LP, Jin J, et al., 2019. An equivalent generating algorithm to model fuzzy Petri net for knowledge-based system. *J Intell Manuf*, 30(4):1831-1842. https://doi.org/10.1007/s10845-017-1355-x

Ziani R, Felkaoui A, Zegadi R, 2017. Bearing fault diagnosis using multiclass support vector machines with binary particle swarm optimization and regularized Fisher's criterion. *J Intell Manuf*, 28(2):405-417. https://doi.org/10.1007/s10845-014-0987-3