Fault diagnosis of PLC-based discrete event systems using Petri nets

Yongyao Li¹, Ya Wang², Guanghui Zhu¹,² *, Li Yin¹ and Huimin Zhang³

Abstract
This paper addresses the fault diagnosis problem of PLC-based systems that can be modeled as Petri nets under a certain level of abstraction. The existing Petri-net-based fault diagnosis approaches often associate transitions and/or places with sensors and require that any change in sensor readings needs to be treated by a PLC, leading to a situation that the PLC would be too busy processing the changes in sensor readings to perform other tasks. This paper assumes that a PLC does not monitor the changes of readings of sensors all the time, but periodically reads the values of sensors when needed. The system output is defined as a marking sequence interleaved with possible observed transitions. A fault diagnosis algorithm is developed by defining and solving integer linear programing (ILP) problems whose size is regardless of the length of the system output. The proposed approach enjoys high computational efficiency compared with other ILP-based approaches and is more suitable for fault diagnosis of PLC-based systems with low computing power.

Keywords
Petri net, fault diagnosis, programmable logic controller, discrete event system

Date received: 6 December 2021; accepted: 11 July 2022

Introduction

Position of this paper
The rapid development of electronic technology has made the computer-integrated systems, which can be abstracted as discrete event systems (DESs),¹ more and more complex, structurally and functionally. Fault diagnosis aims to ensure the safe and stable operation of these systems by detecting and isolating faults as soon as possible (In this paper, “failure” and “fault” have the same meaning and we use them interchangeably. However, they may have slightly different meanings in some publications). More specifically, fault diagnosis requires to complete three tasks: fault detection, fault isolation, and fault identification, corresponding to determining if a system is running normally, revealing the locations and numbers of faults, and identifying the specific nature of faults, respectively.²⁻⁵

In real industrial systems, the control of a system can often be implemented by a programmable logic controller (PLC). In general, a PLC consists of a central processing unit, a storage unit, and an input-output unit, and can receive, send and process various electronic signals. A PLC is connected to the sensors distributed on the components of a system such that it can collect the control parameters of the system by reading these sensors. Many existing fault diagnosis approaches in the literature can be implemented with PLCs.

In the domain of DESs, many existing fault diagnosis approaches have two main disadvantages: (i) a number of approaches⁶⁻⁹ have exponential computational complexity in the worst case and their computational cost may be prohibitive for large systems and (ii) some approaches¹⁰⁻¹² require that any change in sensor readings needs to be treated by a PLC such that the PLC would be too busy processing the changes in sensor readings to perform other tasks.

To overcome these disadvantages, we propose a fault diagnosis approach which equips a system with more sensors and reads sensors at a fixed time interval. The proposed approach has higher computational efficiency and is more appropriate for use in real-world systems.
systems with low computing power. On the other hand, the main drawback of the proposed approach is the higher cost of purchasing sensors and the higher difficulty of deploying them.

Differently from the Petri-net-based approach proposed in this paper, some other approaches based on structural analysis\textsuperscript{13,14} or fault trees\textsuperscript{15,16} are also exposed. In contrast to actively detect and isolate faults in this paper, the stability property of a system is explored in Foroozanfar et al.\textsuperscript{17} and Lutz-Ley and Lopez-Mellado.\textsuperscript{18} A stable system can automatically return to a normal state from a fault state in finite number of steps. Before presenting the details of the proposed approach, we review some typical publications on fault diagnosis of DES.

Some original theoretical results for fault diagnosis are exposed based on automata. In Sampath et al.,\textsuperscript{6} a plant is modeled as an automaton in which faults are represented by unobservable events and each observable event is associated with a sensor. A diagnoser for fault diagnosis is constructed offline by an algorithm with exponential complexity. When an event is observed, by inspecting the diagnoser, the fault state of the plant may be one of three cases: faults do not occur, faults may have occurred, and faults must have occurred. This fault diagnosis approach is applied to a realistic HVAC system in Sampath et al.\textsuperscript{19}

In contrast to automaton models, Petri nets, as a typical mathematical model of DESs, have been widely used in supervisory control,\textsuperscript{20–23} performance optimization,\textsuperscript{24–26} and model identification\textsuperscript{10,11,27,28} because of their intrinsic distributed features and efficiency in handling large systems. Simultaneously, in recent decades, many fault diagnosis approaches based on Petri nets have also been reported, which can be generally categorized into two classes.

In the first class, the sensors are equipped with observable transitions and the output of a system is defined as a sequence of observable transitions. Integer linear programming (ILP) problems often need to be built and solved for performing fault diagnosis. In Dotoli et al.,\textsuperscript{7} an online fault diagnosis approach is reported that builds an ILP problem based on the observed transition sequence. By solving an ILP problem constructed based on the observed evolution, the unobservable transitions are identified that characterize the number and locations of faults.

Ru and Hadjicostis\textsuperscript{12} model systems as partially observed Petri nets (i.e. nets equipped with transition and place sensors) and define the system output as an ordered sequence of three tuples $(M, t, M')$’s, where $M$ and $M'$ are two markings and $t$ is a transition. After transforming the partially observed Petri net into a labeled net, they propose an online fault diagnosis algorithm based on reachability graph analysis of the unobservable subnet of the labeled net. The study in Lefebvre\textsuperscript{34} also utilizes partially observed Petri nets and provides an efficient fault diagnosis algorithm by solving a number of linear matrix inequalities. The number of times of solving inequalities is linear in the length of an observed sequence.

In Ramirez-Trevino et al.,\textsuperscript{35} a bottom-up modeling methodology is explored to build an interpreted Petri net (IPN) model of a system. By dividing the set of places into a subset of places modeling fault states and a subset of places modeling normal states, the diagnosability of the built IPN is formally defined and a scheme for detecting and locating fault states is developed. By assuming that all places are observable, Genc and Lafontaine\textsuperscript{36} extend the diagnoser approach based on automata in Sampath et al.\textsuperscript{6} to the context of Petri nets. A Petri net diagnoser is constructed to perform online diagnosis. This approach is of high computational complexity since the marking enumeration upon observing an transition at each step is needed.

The study in Pencole and Subias\textsuperscript{37} explores the diagnosability of safe labeled time Petri nets based on the notion of event patterns. An efficient approach of determining diagnosability of a safe Petri net is reported by means of model-checking techniques. Al-Ajeli and Parker propose a fault diagnosis approach for labeled Petri nets based on Fourier-Motzkin elimination. A diagnoser in the form of sets of inequalities is first constructed offline and then online fault diagnosis is performed by checking the constructed diagnoser when
observing an event. This approach provides a better trade-off between the size of the diagnoser and diagnosis time. In recent years, some extended versions of fault diagnosis are studied, such as robust diagnosis, diagnosability enforcement, and synchronous diagnosis. Two comprehensive surveys on fault diagnosis can be found in Al-Ajeli and Parker and Lafortune et al.

Contributions
As already mentioned above, the first class of fault diagnosis approaches often needs to solve ILP problems whose size is linear or polynomial with respect to the length of the observed sequence. When the observed sequence is very long, these approaches become computationally prohibitive. On the other hand, the second class of approaches generally assumes that all or part of places are observable, that is, these places are equipped with sensors. Moreover, the approaches in the second class usually require that any change in the readings of place sensors needs to be treated by a PLC. A Petri net model of a real-world system may include a large number of places and there are a deluge of changes of sensor readings during the operation of the system. The PLC would be too busy processing the changes in sensor readings to perform other tasks. Consequently, in some cases, the existing fault diagnosis approaches are not applicable to real-world systems.

In this paper, we assume that part of transitions and all places are equipped with sensors. In addition, due to the limited processing power of a PLC, we assume that it does not monitor the changes of readings of place sensors all the time, but periodically reads the values of the sensors when needed. The system output is defined as a marking sequence interleaved with possible observed transitions.

Based on the observed system output, a fault diagnosis algorithm is developed by defining and solving ILP problems whose size is regardless of the length of the system output. Thus, the computational cost of the algorithm will not increase with the increase of the length of the system output. Namely, the algorithm enjoys a high computational efficiency for those systems that satisfy the assumptions made in this paper. On the other hand, since the PLC only requires to read sensor when needed. The system output is defined as a marking sequence interleaved with possible observed transitions. The set of transition sequences that are enabled at the initial marking $M_0$ is defined as

$$M_0 = M + C \cdot \pi(\sigma),$$

where $\pi$ is a function, defined by $\pi : T^* \rightarrow \mathbb{N}^p$, which computes the Parikh vector of a transition sequence $\sigma$. If a transition $t$ is in $\sigma$, we write $t \in \sigma$ for simplicity.

The set of transition sequences that are enabled at the initial marking $M_0$ is defined as

$$\mathcal{L}(N, M_0) = \{ \sigma \in T^* | M_0[\sigma] \}. $$

The reachability set, represented as $R(N, M_0)$, of a net system contains all markings that are reachable from $M_0$ by firing a transition sequence $\sigma \in \mathcal{L}(N, M_0)$, that is,$ R(N, M_0) = \{ M \in \mathbb{N}^m | (\exists \sigma \in T^*) M_0[\sigma] = M \}.$

A transition is said to be observable if it is equipped with a sensor that monitors its firing; otherwise, it is called unobservable. We denote by $T_0$ and $T_u$ the set of observable transitions and the set of unobservable transitions, respectively. In addition, the cardinalities of $T_0$ and $T_u$ are represented as $n_0$ and $n_u$, respectively. Thus,
the set $T$ of transitions is partitioned into two disjoint subsets $T_o$ and $T_u$ with $T = T_o \cup T_u$. A place of a Petri net is said to be measurable if the number of tokens residing into it can be detected by a sensor. In this paper, we assume that all places of a Petri net are measurable. A Petri net is said to be acyclic if it does not contain any directed cycle.

**Theorem 1.** A marking $M \in \mathbb{N}^n$ is reachable from $M_0$ in an acyclic Petri net system $(N, M_0)$ if and only if there exists a column vector $x \in \mathbb{N}^n$ such that $M = M_0 + C \cdot x$, where $m$ and $n$ are the numbers of places and transitions, respectively.

**Definition 1.** Given a Petri net $N = (P, T, Pre, Post)$ and a transition subset $T' \subseteq T$, the $T'$-induced subnet of $N$ is defined as $N_{T'} = (P, T', Pre_{T'}, Post_{T'})$, where $Pre_{T'}$ and $Post_{T'}$ are the restrictions of $Pre$ and $Post$ on $P \times T'$, respectively.

According to Def. 1, it is straightforward to obtain the $T_o$-induced subnet of $N = (P, T, Pre, Post)$, denoted by $N_o = (P, T_o, Pre_o, Post_o)$. We also call the $T_u$-induced subnet the unobservable subnet of $N$. The incidence matrix of $N_o$ is represented by $C_o = Post_o - Pre_o$. Unless otherwise stated, the unobservable subnet of the considered Petri nets in this paper is assumed to be acyclic.

To characterize failures in a real-world system, the unobservable transitions in $T_u$ are further divided into regular unobservable transitions, denoted by $T_{reg}$, and fault transitions, denoted by $T_f$, that is, $T_u = T_{reg} \cup T_f$ and $T_{reg} \cap T_f = \emptyset$. The fault transition set $T_f$ is partitioned into $r$ fault classes, that is, $T_f = \bigcup_{j=1}^r T_f^j$, and the partition is represented as $\Pi_f = \{T_f^1, \ldots, T_f^r\}$. For simplicity, we say that $T_f$ occurs if a fault $f \in T_f^i$ takes place in a system. Given a sequence $\sigma \in T^*$, we use $t \sigma$ to denote that $t$ is contained in $\sigma$ and $T_f^i \sigma$ to denote that there exists at least a transition $f \in T_f^i$ such that $f \sigma$.

A Petri net system with a labeling function $\lambda : T \cup \{e\} \to E \cup \{\epsilon\}$ is called a labeled Petri net (LPN) system, denoted by $(N, M_0, E, \lambda)$, where $E$ is an alphabet and $\epsilon$ represents the empty string. The label of the empty string $e$ is itself, that is, $\lambda(e) = \epsilon$. The set of transitions that have the same label $e$ is denoted by $T_e = \{t \in T | \lambda(t) = e\}$. The labeling function $\lambda$ is extended to a transition sequence $\sigma = t_1 t_2 \ldots t_h \in T^*$ with $h$ being its length by defining $\lambda(\sigma) = \lambda(t_1)\lambda(t_2)\ldots\lambda(t_h)$. We obtain the underlying Petri net of an LPN by removing all labels of transitions of the LPN.

**Problem definition**

In general, faults in a system can be divided into three types: incipient, intermittent, and permanent. The approach proposed in this paper is based on Petri nets (a typical discrete event model) and are appropriate for faults that cause a distinct change in the state of system components but do not necessarily bring the system to a halt. Thus, the proposed approach can be used to detect intermittent or permanent faults. These faults may originate from different system components, such as sensors, actuators, or controllers.

The problem of fault diagnosis consists in determining whether faults have occurred according to the behavior of a system. In general, the behavior of a system is described by its output, that is, sensor-readings at each step during the system evolution.

This paper deals with the fault diagnosis problem of PLC-based Petri nets. Though all places of the considered net are measurable, we assume that a PLC does not monitor the changes of readings of place sensors all the time due to the limited processing power of the PLC. To define the output of a Petri net system, the following assumptions hold for the systems under consideration:

(A1) Reading place sensors and firing any transition can be done instantly.

(A2) Place sensors are read once in $K$ time units.

In a real PLC-controlled system, reading sensors is usually implemented by a hardware interrupt and performing an action (i.e., firing a transition) is usually carried out by a callback function of programing languages. In this way, the controller of the system can know if the operation of reading sensors or performing an action is successful and obtain the corresponding sensor data if the answer is positive. However, in this paper, we mostly focus on the logic steps of a fault diagnosis algorithm, without explicitly considering its software implementation. Thus, we make Assumption (A1) for the convenience of discussion. Simultaneously, we make Assumption (A2) to avoid a situation that the PLC of a system is too busy processing the changes of readings of place sensors to perform other tasks.

On the basis of Assumptions (A1) and (A2), the output of a system at each step is represented by two possible cases:

1. A marking pair $(M, M')$ with $M, M' \in R(N, M_0)$.
2. A three tuple $(M, t_o, M')$ with $M, M' \in R(N, M_0)$ and $t_o \in T_o$.

In the first case, $M'$ is reachable from $M$ by firing an unobservable transition sequence $\sigma_o \in T_o^*$, that is, $M[\sigma_o]M'$, and $M'$ is obtained by reading place sensors after $K$ time units from the starting point $M$.

In the second case, starting from $M$, an observable transition $t_o$ fires before reaching $K$ time units and the output is represented as $(M, t_o, M')$, satisfying $M[\sigma_o]M_o[t_o]M'$, where $\sigma_o \in T_o$ and $M_o \in R(N, M_0)$. Namely, marking $M_o$ is reached from $M$ by firing an unobservable transition sequence. After observing transition $t_o$, the timer is reset to zero and the elapsed time units are recounted. Subsequently, we provide an example to clarify this.
Example 1. Consider the Petri net shown in Figure 1, where \( T_u = \{ t_1 \} \), \( T_s = \{ f_1, f_2, f_3, f_4, u_1, u_2, u_3 \} \), and the initial marking is \( M_0 = p_1 + p_2 \). If the time interval to read place sensors is four time units (i.e., \( K = 4 \)), then a possible evolution of the net is shown in Figure 2. The timeline shows the moments to read place sensors. The output sequence obtained along this timeline is represented as a marking-transition sequence, that is, \( M_0 M_1 t_1 M_2 M_3 t_2 M_4 M_5 \), which contains five steps \((M_0, M_1), (M_1, t_1, M_2), (M_2, M_3), (M_3, t_2, M_4)\) and \((M_4, M_5)\), as shown in the lower part of Figure 2.

In Step 1, from the starting point \( M_0 \), no observable transition is observed during four time units. Then, at the fourth time unit, place sensors are read and the marking \( M_1 = p_2 + p_3 \) is obtained. Thus, the output in Step 1 is the first case mentioned above, that is, a marking pair \((M_0, M_1)\). In Step 2, a timer restarts from marking \( M_1 \) and we observe transition \( t_1 \) before four time units elapse. In addition, by reading place sensors, the reached marking after firing \( t_1 \) is \( M_2 = p_3 + p_4 \). Thus, the output in Step 2 can be represented as the second cases defined previously, that is, a three tuple \((M_1, t_1, M_2)\). The outputs in Steps 3–5 can be similarly explained and are illustrated in Figure 2.

Definition 2. A trace of a Petri net system \((N, M_0)\) with \( T = T_u \cup T_s \) is a sequence \( \sigma = M_0 b_1 M_2 b_2 M_3 b_3 ... b_L M_L \) such that there exist unobservable transition sequences \( \sigma_{u_1}, \sigma_{u_2}, ..., \sigma_{u_L} \in T_u \) satisfying \( M_0[\sigma_{u_1} b_1] M_1 ... \sigma_{u_L} b_L M_L \), where \( L \geq 1 \) is a positive integer, \( b_i \in T_u \cup \{ \varepsilon \} \), \( M_i \in N(N, M_0) \), \( i = 1, 2, ..., L \).

The set of traces of a net system \((N, M_0)\) is defined as \( T(N, M_0) = \{ \sigma | \sigma \text{ is a trace} \} \). Obviously, the output sequence of a Petri net in this paper, defined under Assumptions (A1) and (A2), is a trace. For example, the output sequence, say \( \sigma_1 \), in Example 1 is a trace and it holds \( \sigma_1 \in T(N, M_0) \). Now, we are ready to formally define the fault diagnosis problem addressed in this paper.

Problem 1. Given a Petri net system \((N, M_0)\) and a partition \( \Pi_f = \{ T_f, ..., T_f \} \) of \( r \) fault classes, the problem consists in determining the occurrence of each fault class \( T_f \) (\( i = 1, 2, ..., r \)) till observing a trace \( \sigma = M_0 b_1 M_1 b_2 M_2 ... b_L M_L \) that contains multiple steps with each being in the form of a marking pair \((M, M')\) or a three tuple \((M, t, M')\).

Fault diagnosis algorithm for Petri nets

In this section, we define a local diagnoser and a global diagnoser for solving Problem 1 and develop a fault diagnosis algorithm based on integer linear programming techniques. To formally present the algorithm, several new definitions are first provided.

Definition 3. Given a net system \((N, M_0)\) with \( T = T_u \cup T_s \) and two markings \( M, M' \in N(N, M_0) \), the set of unobservable transition sequences whose firings bring the system from \( M \) to \( M' \) is defined by \( \Lambda(M, M') = \{ \sigma_u \in T_u | M[\sigma_u] M' \} \).

It is clear that if \( M' \) is reachable from \( M \) by firing a sequence \( \sigma_u \in T_u \) that is, \( M'[\sigma_u] M' \), then \( \Lambda(M, M') \neq \emptyset \). When observing a pair \((M, M')\) during the evolution of a step of a system, the following local diagnoser describes the diagnosis decision of each fault class \( T_f \) after observing \((M, M')\).

Definition 4. A local diagnoser is a function \( \Delta : \mathbb{N}^r \times \mathbb{N}^r \times \Pi_f \to \{ 0, 1, 2 \} \) which associates a pair \((M, M')\) and each fault class \( T_f \in \Pi_f \) with a diagnosis state such that

- \( \Delta(M, M', T_f) = 0 \) if for all \( f \in T_f \) and for all \( \sigma_u \in \Lambda(M, M') \) it holds \( f \notin \sigma_u \), that is, there are multiple paths from \( M \) to \( M' \) in the reachability graph but each path does not contain transition \( f \). In such a case, the system behaves normally between \( M \) and \( M' \) and no fault occurs.
- \( \Delta(M, M', T_f) = 1 \) if there exist two unobservable transition sequences \( \sigma_u, \sigma_w \in \Lambda(M, M') \) such that (i) there exists a fault \( f \in T_f \) satisfying \( f \in \sigma_u \), and (ii) for all \( f \in T_f \), it holds \( f \notin \sigma_w \), that is, there are at least two possible paths from \( M \) to \( M' \) in the reachability graph, one passing a fault \( f \in T_f \) but another passing none of them.
- \( \Delta(M, M', T_f) = 2 \) if for all \( \sigma_u \in \Lambda(M, M') \) there exists a fault \( f \in T_f \) satisfying \( f \in \sigma_u \), that is, each path from \( M \) to \( M' \) in the reachability graph contains a fault \( f \in T_f \).
Theorem 2. Let \( \langle N, M_0 \rangle \) be a Petri net system with \( T = T_o \cup T_u \) and \( T_f = T_1 \cup \ldots \cup T_f \subseteq T_u \). For two markings \( M, M' \in R(N, M_0) \) with \( M' \) being reachable from \( M \) by firing an unobservable transition sequence, and a fault class \( T_f' \in \Pi_f \), it holds

\[
\Delta(M, M', T_f') = 2 \quad \text{if and only if ILPP 1 has no feasible solution,}
\]

\[
\Delta(M, M', T_f') = 0 \quad \text{if and only if ILPP 2 has no feasible solution,}
\]

\[
\Delta(M, M', T_f') = 1 \quad \text{if and only if both ILPP 1 and ILPP 2 admit a solution,}
\]

Proof. We prove the results (1)–(3) in turn.

(1): (if) Since \( M' \) is reachable from \( M \) by firing an unobservable transition sequence, there necessarily exists a column vector \( y \) satisfying \( M' = M + C_u \cdot y \). On the other hand, if ILPP 1 has no feasible solution, then there does not exist an unobservable transition sequence that contains none of fault transitions in \( f \in T_f \) and whose firing brings the system from \( M \) to \( M' \), that is, for all \( \sigma_u \in \Lambda(M, M') \), \( T_f' \notin \sigma_u \). According to Def. 4, we have \( \Delta(M, M', T_f') = 2 \). (only if) If ILPP 1 admits a solution, there exists a sequence \( \sigma_u \in T_u \) such that \( M[\sigma_u]M' \) and \( T_f' \notin \sigma_u \) according to Theorem 1. Namely, it holds \( \Delta(M, M', T_f') = 1 \) or 0. Clearly, this violates the condition \( \Delta(M, M', T_f') = 2 \). Thus, this result is proved by contradiction.

(2): (if) If ILPP 2 has no feasible solution, all possible unobservable transition sequences from \( M \) to \( M' \) in the reachability graph do not pass one of fault transition \( f \in T_f \), that is, for all \( \sigma_u \in \Lambda(M, M') \), \( T_f' \notin \sigma_u \). Thus, we conclude \( \Delta(M, M', T_f') = 0 \) by Def. 4. (only if) We prove this result by contradiction. If ILPP 2 has a solution, by Theorem 1, there is a sequence \( \sigma_u \) satisfying \( M[\sigma_u]M' \) and \( T_f' \notin \sigma_u \), that is, \( \Delta(M, M', T_f') = 0 \). This violates the condition \( \Delta(M, M', T_f') = 2 \).

(3): (if) If both ILPP 1 and ILPP 2 have a solution, by Theorem 1, there exist two sequence \( \sigma_{u_1}, \sigma_{u_2} \) such that \( T_f' \notin \sigma_{u_1} \) and \( T_f' \notin \sigma_{u_2} \). Thus, it holds \( \Delta(M, M', T_f') = 1 \) by Def. 4. (only if) By contradiction, we first assume that ILPP 1 has no solution. Then, we have \( \Delta(M, M', T_f') = 2 \) according to the result (1). This violates the condition \( \Delta(M, M', T_f') = 1 \). Thus, ILPP 1 necessarily has a solution. If we assume that ILPP 1 has no solution, it holds \( \Delta(M, M', T_f') = 0 \) according to the result (2). This violates \( \Delta(M, M', T_f') = 1 \). Thus, we prove that both ILPP1 and ILPP1 admit a solution.

\[\square\]

Given two markings \( M, M' \in R(N, M_0) \) and a fault class \( T_f' \in \Pi_f \), the local diagnoser \( \Delta(M, M', T_f') \) can be computed by solving ILPPs 1 and 2 according to Theorem 2. However, the final aim of fault diagnosis is to determine the occurrence of faults till observing a trace \( \sigma = M_0b_1M_1b_2M_2\ldots b_iM_L \), as described in
Problem 1. Thus, we provide the following definitions to achieve the aim.

Definition 5. Given a trace \( \sigma = M_0 b_1 M_1 b_2 M_2 \ldots b_L M_L \in T(N, M_0) \) of a net system \( \langle N, M_0 \rangle \), the set of its associated marking pairs is defined as

\[
\mathcal{M}_\sigma = \{(M_i, M)| M = M_{i+1} - C \cdot \pi(b_{i+1}) \land M \neq M_i, i = 0, \ldots, L-1\},
\]

where \( \pi(b_i) = \overline{b}_i \) if \( e_i = \varepsilon \).

Example 3. Consider the net shown in Figure 1 and one of its traces \( \sigma = M_0 b_1 M_1 b_2 M_2 \ldots b_L M_L \) as shown in Figure 4. In this trace, we have \( b_1 = b_3 = \varepsilon \) and \( b_2 = b_4 = t_1 \). In addition, it is clear that \( C \cdot \pi(t_1) = [00 \ 110]^T \). Thus, by Def. 5, it holds

\[
\mathcal{M}_\sigma = \{(p_1 + p_2, p_2 + p_3), (p_2 + p_3, 2p_3), (p_3 + 2p_4, 2p_3), (p_3 + 2p_4 - p_2 + p_3)\}.
\]

When computing the set \( \mathcal{M}_\sigma \), we delete a marking pair (2p_3, 2p_3) since \( 2p_3 = p_3 \) (i.e. \( M = M_i \)).

Definition 6. A global diagnoser of a net \( \langle N, M_0 \rangle \) is a function \( \Theta : T(N, M_0) \times \Pi_f \rightarrow \{0, 1, 2\} \) that associates a trace \( \sigma \in T(N, M_0) \) and a fault class \( T' \in \Pi_f \) with a diagnosis decision such that

- \( \Theta(\sigma, T') = 0 \) if for all pairs \( (M, M') \in \mathcal{M}_\sigma \), it holds \( \Delta(M, M', T') = 0 \).
- \( \Theta(\sigma, T') = 1 \) if (i) there exists a pair \( (M, M') \in \mathcal{M}_\sigma \) such that \( \Delta(M, M', T') = 1 \) and (ii) there does not exist a pair \( (M, M') \in \mathcal{M}_\sigma \) such that \( \Delta(M, M', T') = 2 \).
- \( \Theta(\sigma, T') = 2 \) if there exists a pair \( (M, M') \in \mathcal{M}_\sigma \) such that \( \Delta(M, M', T') = 2 \).

In plain terms, the diagnosis decision \( \Theta(\sigma, T') = 0 \) means that the fault class \( T' \) does not occur during each step of the trace \( \sigma \). The decision \( \Theta(\sigma, T') = 2 \) implies that \( T' \) necessarily has occurred in one of steps of \( \sigma \). Consequently, all paths from \( M_0 \) to \( M_L \) in the reachability graph include the transitions in the step and necessarily pass a fault transition \( f \in T' \).

Algorithm 1: Fault diagnosis of a trace in a Petri net system

Input: A trace \( \sigma = M_0 b_1 M_1 b_2 M_2 \ldots b_L M_L \) in a net system \( \langle N, M_0 \rangle \)
Output: Diagnosis decision \( \Theta(\sigma, T') \) of each fault class \( T' \in \Pi_f \)

1. \( \Theta = \overline{0} \in \{0, 1, 2\}^f \)
2. Compute the set \( \mathcal{M}_\sigma \) of marking pairs associated with \( \sigma \) according to Def. 5.
3. for \( i = 1, \ldots, r \) do
4. Consider the fault class \( T' \)
5. \( D = 0; \)
6. for all \( (M, M') \in \mathcal{M}_\sigma \) do
7. \( d = 0; \)
8. Build and solve ILPP 1, i.e., \( \Gamma_1(M, M', T') \);
9. if ILPP 1 has no feasible solution then
10. \( d = 2; \)
11. else
12. Build and solve ILPP 2, i.e., \( \Gamma_2(M, M', T') \);
13. if ILPP 2 has no feasible solution then
14. \( d = 0; \)
15. else
16. \( d = 1; \)
17. \( D = \max(D, d); \)
18. \( \Theta(i) = D; \)
19. Output \( \Theta; \)

In Def. 6, we observe that the global diagnoser of a trace \( \sigma \) can be computed by inspecting all local diagnosers of steps in the trace. Next, we develop an algorithm to detect faults that have occurred till the observation of a trace.

The main logical flow of Algorithm 1 is illustrated by the flowchart shown in Figure 5. More specifically, in Line 1, the global diagnoser \( \Theta \) is initialized to an \( r \)-dimensional column vector \( \Theta = \overline{0} \in \{0, 1, 2\}^r \) such that \( \Theta(i) \) stands for the diagnosis decision of fault class \( T'_i \), where \( r \) is the number of fault classes and 0, 1, and 2 are the diagnosis decisions.

The main computational cost of Algorithm 1 stems from the solution of ILPP 1 and/or ILPP 2. It is well known that the complexity of an integer linear programming problem is NP-complete and is closely related to its size (i.e. the numbers of unknown variables and constraints). However, in practice, many integer linear programming problems can be efficiently solved using commercial solvers (such as Gurobi Solver used in the paper).

Since integer linear programming problems ILPP 1 and ILPP 2 have the same size, we analyze the size of ILPP 1 only. The number of unknown variables is denoted by...
Algorithm 6 outputs the correct diagnosis decision $d = 0$, then the algorithm outputs.

This is obviously correct according to Def. 6. On the other hand, if there exists a pair $(M, M') \in \mathcal{M}_\sigma$ with $d = \Delta(M, M', T_f) = 1$ and there does not exist a pair $(M, M')$ with $\Delta(M, M', T_f) = 2$, the maximal value will be $1$, that is, $D = \max(D, d) = 1$. This result can be readily verified by Def. 6. Finally, if there exists a pair $(M, M') \in \mathcal{M}_\sigma$ with $d = \Delta(M, M', T_f) = 2$, it is clear that $D = \max(D, d) = 2$. This is also consistent with Def. 6.

$\square$

**Example 4.** Let us consider the net system shown in Figure 6, where $T_\sigma = \{t_1, t_2\}$, $T_u = \{u_1, u_2, f_1, f_2\}$, $T_f^1 = \{f_1\}$, and $T_f^2 = \{f_2\}$. Assume that one of the traces of the net is

$$\sigma = M_0 t_1 M_1 t_2 M_3 \sigma,$$

where $n_u$ is the number of unobservable transitions; the number of constraints is denoted by

$$\mathcal{J} = m + 1,$$

where $m$ is the cardinality of place set. We observe that the size of ILPP 1 is regardless of the length $L$ of a trace $\sigma$.

**Proposition 1.** Given a trace $\sigma = M_0 b_1 M_1 b_2 M_2 \ldots b_L M_L$ of a net system $(N, \mathcal{M}_0)$, for all fault classes $T_f^i \in \Pi_f$ Algorithm 6 outputs the correct diagnosis decision $\Theta(\sigma, T_f^i)$.

**Proof.** According to Theorem 2, the local diagnosis decision $\Delta(M, M', T_f^i)$ of each marking pair $(M, M') \in \mathcal{M}_\sigma$ is correctly computed in Lines 6–16. We next prove that it is correct to compute the global diagnosis decision by selecting the maximal local diagnosis decision, that is, $D = \max(D, d)$ in Line 17. If for all $(M, M') \in \mathcal{M}_\sigma$, we have $\Delta(M, M', T_f^i) = 0$, that is, $d = 0$, then the algorithm outputs $D = \Theta(\sigma, T_f^i) = 0$. This is obviously correct according to Def. 6. On the other hand, if there exists a pair $(M, M') \in \mathcal{M}_\sigma$ with $d = \Delta(M, M', T_f^i) = 1$ and there does not exist a pair $(M, M')$ with $\Delta(M, M', T_f^i) = 2$, the maximal value will be $1$, that is, $D = \max(D, d) = 1$. This result can be readily verified by Def. 6. Finally, if there exists a pair $(M, M') \in \mathcal{M}_\sigma$ with $d = \Delta(M, M', T_f^i) = 2$, it is clear that $D = \max(D, d) = 2$. This is also consistent with Def. 6. $\square$

**Figure 6.** A Petri net for Example 4.

$$I = n_u,$$

where $n_u$ is the number of unobservable transitions; the number of constraints is denoted by

$$\mathcal{J} = m + 1,$$

where $m$ is the cardinality of place set. We observe that the size of ILPP 1 is regardless of the length $L$ of a trace $\sigma$. We compute the global diagnoser $\Theta(\sigma, T_f^1)$ and $\Theta(\sigma, T_f^2)$ using Algorithm 1. The global diagnosis decisions are represented as a two-dimensional column vector $\Theta$ such that $\Theta(1) = \Theta(\sigma, T_f^1)$ and $\Theta(2) = \Theta(\sigma, T_f^2)$. At first, $\Theta$ is set to $\Theta = [00]^T$. According to Def. 5, we have $\mathcal{M}_\sigma = \{P_1, P_2, P_3, P_4\} = \{2p_1 + 2p_6, p_1 + p_2, 2p_5 + p_6 + p_2 + 2p_8\}$. $\Theta$ is updated by $\Theta(1) = \max(\Theta(1), d) = 0$. Using the

\[
\begin{bmatrix}
2 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 \\
2 & 1 & 2 & 1 & 2
\end{bmatrix}
\]

We compute the global diagnoser $\Theta(\sigma, T_f^1)$ and $\Theta(\sigma, T_f^2)$ using Algorithm 1. The global diagnosis decisions are represented as a two-dimensional column vector $\Theta$ such that $\Theta(1) = \Theta(\sigma, T_f^1)$ and $\Theta(2) = \Theta(\sigma, T_f^2)$. At first, $\Theta$ is set to $\Theta = [00]^T$. According to Def. 5, we have $\mathcal{M}_\sigma = \{P_1, P_2, P_3, P_4\} = \{2p_1 + 2p_6, p_1 + p_2, 2p_5 + p_6 + p_2 + 2p_8\}$. $\Theta$ is updated by $\Theta(1) = \max(\Theta(1), d) = 0$. Using the

\[
\begin{bmatrix}
2 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 \\
2 & 1 & 2 & 1 & 2
\end{bmatrix}
\]
same reasoning, we have \( \Theta(2) = \max(\Theta(2), 0) = 0 \). For the pair \( P_2 \in M_r \), we have \( \Theta(1) = \max(\Theta(1), 1) = 1 \) and \( \Theta(2) = \max(\Theta(2), 1) = 1 \). Analogously, for \( P_3 \), the local diagnosis decision is \( d = [\Delta(M, M', T_1) \Delta(M, M', T_2)]^T = [1 2]^T \) and the global diagnosis decision is updated by \( \Theta = \max(\Theta(1), 1)\max(\Theta(2), 2) = [1 2]^T \). Finally, for \( P_4 \), we have \( d = [2 2]^T \) and \( \Theta = \max(\Theta(1), 2) \max(\Theta(2), 2) = [2 2]^T \).

Fault diagnosis algorithm for LPNs

This section extends the fault diagnosis algorithm for Petri nets to the case of labeled Petri nets. In LPNs, two or more observable transitions may share the same label and an observation often corresponds to multiple traces of the underlying Petri nets. Thus, the algorithm for LPNs needs to enumerate all possible transitions with the same label and combine diagnosis results for different traces.

Definition 7. A labeled trace of an LPN system \((N, M_0, E, \lambda)\) is a sequence \( \omega = M_0e_1M_1e_2M_2 \cdots e_r M_L \) such that there exists a trace \( \sigma = M_0b_1M_1b_2M_2 \cdots b_l M_L \in T(N, M_0) \), where \( L \geq 1 \) is a positive integer, \( e_i \in E \cup \{ \varepsilon \} \), \( b_i \in T_\sigma \cup \{ \varepsilon \} \), and \( M_i \in R(N, M_0) \), \( i = 1, \ldots, L \).

In plain words, if we observe a labeled trace \( \omega \), the underlying evolution of a system is a trace \( \sigma \) such that \( \lambda(b_i) = e_i \) with \( i = 1, \ldots, L \). We use \( e_i M_i \in \omega \) to denote that \( e_i M_i \) is contained in \( \omega \), \( i \in \{ 1, \ldots, L \} \). To conveniently present the fault diagnosis algorithm for LPNs, we define the following operator.

Definition 8. The operator \( \otimes: \{ 0, 1, 2 \} \times \{ 0, 1, 2 \} \to \{ 0, 1, 2 \} \) associates two operands \( a, b \in \{ 0, 1, 2 \} \) with a value \( c \in \{ 0, 1, 2 \} \) according to Table 1, which is written as \( a \otimes b = c \).

Given a labeled trace \( \omega \), there often exist multiple traces \( \sigma \)'s that have the same observation. We use the operator \( \otimes \) to combine the diagnosis decisions of these traces to form the diagnosis result of the labeled trace. Algorithm 2 illustrates the usage of \( \otimes \). Before present the algorithm, we define an ILP problem:

\[
\Gamma(M, M') = \begin{cases} 
M' = M + C_u \cdot y & \text{if } y \in \mathbb{N}^n, \\
\end{cases}
\]

which characterizes if there exists an unobservable transition sequence \( \sigma_u \) such that \( M'[\sigma_u]M' \) and \( y = \pi(\sigma_u) \). If \( \Gamma(M, M') \) admits a solution, there necessarily exists such a sequence by Theorem 1. Otherwise, such a sequence does not exist. Now, it is ready to present the fault diagnosis algorithm for LPNs.

In Algorithm 2, the global diagnosis decisions of a labeled trace \( \omega \) are stored in an \( r \)-dimensional vector \( \Phi \), as shown in Line 1. All steps \( e_i M_i \in \omega \) \( (i = 1, \ldots, L) \) are considered in turn in Line 3. For each \( e_i M_i \), the local diagnosis decisions are recorded in \( D = \overline{\varepsilon} \in \{ 0, 1, 2 \} \). If \( e_i \) is the empty string in Line 6, we compute the diagnosis decision \( d = \Delta(M, M', T_j) \) of each fault class \( T_j \) by solving ILPP 1 and/or ILPP 2; otherwise, we check all transitions \( T_\sigma \)'s with label \( e_i \) in Line 12 and update \( M' \) by \( M' = M_0 - C \cdot \pi(e_i) \). If the programing \( \Gamma(M, M') \) has a solution in Line 14, there necessarily exists an unobservable transition sequence \( \sigma_u \) such that \( M'[\sigma_u]M' \) and \( y = \pi(\sigma_u) \). Then, the diagnosis decision \( d = \Delta(M, M', T_j) \) can be computed by solving ILPPs 1 and 2 and the diagnosis decisions for different \( T_\sigma \)'s are combined by the operator \( \otimes \) in Line 17. Finally, the global
diagnosis $\Phi$ are obtained by combining diagnosis vector $D$'s of all steps $(e_i,M_i)$'s using the max operation in Line 18.

Example 5. Let us consider an LPN $\langle N, M_0, E, \lambda \rangle$ with $T = T_a \cup T_u$, $T_a = \{t_1, t_2, t_3, t_4\}$, and $E = \{a, b\}$. The labels of all unobservable transitions are the empty string and the labels of observable transitions are represented as $\lambda(t_1) = \lambda(t_2) = a$ and $\lambda(t_3) = \lambda(t_4) = b$ (i.e. $T_a = \{t_1, t_2\}$ and $T_b = \{t_3, t_4\}$). Given a labeled trace $\omega = M_0 a M_1 M_2 b M_3$, we show how Algorithm 2 works by taking $\omega$ as an input. The fault diagnosis process is illustrated by Figure 7.

For Step 1 “ $aM_1$ ” of $\omega$, we consider $t_1, t_2 \in T_a$ in turn, as shown in Line 12. In Figure 7, for both $t_1$ and $t_2$, $\Gamma(M, M')$ admits a solution. The corresponding diagnosis decisions are denoted by $D_1 = [d_1, d_2]$ and $D_2 = [d_3, d_4]$, respectively. According to Line 17, the diagnosis decision vector $D$ for Step 1 is obtained by combining $D_1$ and $D_2$ using the operator $\otimes$, that is, $D = [d_1 \otimes d_3, d_2 \otimes d_4]$, as shown in Figure 7. For Step 2 “ $eM_2$ ”, the condition in Line 6 holds and the diagnosis vector in this step is represented by $D = [d_5, d_6]$. Similar to Step 1, two observable transitions $t_3$ and $t_4$ with the same label $b$ in Step 3 “ $bM_3$ ” are inspected in turn. For $t_3 \in T_u$, the programming $\Gamma(M, M')$ has a solution. However, it has no feasible solution for $t_4 \in T_b$ in Figure 7. We mark this case with a symbol “ $\times$ ” in Figure 7. Finally, the global diagnosis result vector $\Phi$ is obtained by combining all local diagnosis vectors of Steps 1–3 using max operation in Line 18.
Table 2. Nine events of production line 1 and their corresponding transitions.

| Event no. | Transition | Explanation |
|-----------|------------|-------------|
| 1         | $t_1$      | R1 takes a part from II |
| 2         | $u_1$      | R1 puts a part into M1 |
| 2’        | $u_2$      | R1 puts a part into M2 |
| 3         | $t_2$      | R3 takes a part from M1 |
| 3’        | $u_3$      | R3 takes a part from M2 |
| 4         | $t_4$      | R3 puts a part into B1 |
| 5         | $u_4$      | R3 takes a part from B1 |
| 6         | $t_5$      | R3 puts a part into M3 |
| 7         | $t_6$      | R2 takes a part from M3 |
| 8         | $u_5$      | R2 puts a part into AGV1 |
| 9         | $t_7$      | AGV1 transfers a part to O1 |

Table 3. Explanations of some places of the net shown in Figure 9.

| Place | Explanation | Token(s) in the place |
|-------|-------------|-----------------------|
| $p_1$ | entrance II | number of raw part in II |
| $p_{11}$ | entrance I2 | number of raw part in I2 |
| $p_{20}$ | machine M1 | availability of M1 |
| $p_{21}$ | machine M2 | availability of M2 |
| $p_{22}$ | buffer B1 | number of free positions in B1 |
| $p_{23}$ | machine M3 | availability of M3 |
| $p_{24}$ | AGV1 | availability of AGV1 |
| $p_{25}$ | machine M4 | availability of M4 |
| $p_{26}$ | buffer B2 | number of free positions in B2 |
| $p_{27}$ | machine M4 | availability of M5 |
| $p_{28}$ | AGV1 | availability of AGV1 |
| $p_{29}$ | robotic arm R1 | availability of R1 |
| $p_{30}$ | robotic arm R2 | availability of R2 |
| $p_{31}$ | robotic arm R3 | availability of R3 |
| $p_{32}$ | robotic arm R4 | availability of R4 |

Case study

In this section, we show the application and computational efficiency of the proposed approach by means of a plant example. Let us consider the plant shown in Figure 8, which contains five machines (M1–M5), four robotic arms (R1–R4), two buffers with capacity 4 (B1 and B2), two entrances (I1 and I2), two exits (O1 and O2), and two automatic guided vehicles (AGV1 and AGV2).

This plant has two production lines, as shown by the left part and right part of Figure 8, respectively:

- In production line 1 (i.e. the left part of Figure 8), robotic arm R1 first takes a raw part from entrance I1 and then load it into machine M1 or M2. After machine M1 or M2 finishes processing the part, robotic arm R3 unloads the part and puts it into buffer B1. When buffer B1 is not empty, R3 takes a part from B1 and loads it into machine M3. Robotic arm R2 unloads the finished part from M3 and puts it into AGV1 that transfers the part to exit O1.

- In production line 2 (i.e. the right part of Figure 8), robotic arm R1 takes a raw part from entrance I2 and loads it into machine M4. Robotic arm R4 unloads the processed part from M4 and puts it into buffer B2. When B2 is not empty, R4 takes a part from B2 and puts it into machine M5. Robotic arm R2 takes the finished part from M5 and puts it into AGV2. Finally, the part is transferred to exit O2 by AGV2.

Each production line contains nine events, as indicated by the numbers 1–9 in Figure 8. The Petri net model of the plant is shown in Figure 9. Nine events of production line 1 and their corresponding transitions in Figure 9 are demonstrated in Table 2. The events of production line 2 and the corresponding transitions can be analogously explained. Apart from the events in each production line, the components (machines, robotic arms, etc.) of the plant shown in Figure 8 are also explicitly characterized by the places of the Petri net model shown in Figure 9, as listed in Table 3.

We assume that there are two failures in the plant, which are described by two fault transitions $f_1$ and $f_2$ in Figure 9, respectively. Transition $f_1$ characterizes a failure that a part is stored into buffer B1 before being machined by machine M1 and $f_2$ describes that a part is directly processed by machine M5 before being stored into buffer B2.

The labeled Petri net model in Figure 9 has 32 places and 32 transitions, where $T_0 = \{t_1, \ldots, t_{12}\}$, $T_1 = \{f_1, f_2, u_1, \ldots , u_8\}$, $T_2 = \{f_1\}$, and $T_3 = \{f_2\}$. The initial marking is $M_0 = 2p_1 + 2p_{11} + p_{21} + 4p_{21} + 2p_{23} + p_{25}$ + $4p_{20} + 2p_{27} + p_{28} + p_{30} + p_{31} + p_{32}$. Given an alphabet $E = \{a, b, c, d, g\}$, the labels of the observable transitions are represented as $T_a = \{t_1, t_9\}$, $T_b = \{t_2, t_3, t_9\}$, $T_c = \{t_4, t_{10}\}$, $T_d = \{t_5, t_6, t_{11}\}$, and $T_g = \{t_7, t_{12}\}$. In Figure 9, the label of each observable transition is demonstrated in parenthesis beside the transition.

In this case study, we compare the computational efficiency of the proposed approach with the ones in Fanti et al. and Zhu et al. by following the steps below:

**Step 1**: Simulate the operation of the net system shown in Figure 9 and generate an evolution, that is, a sequence that records all reached markings and fired transitions in order.

**Step 2**: Compute three observations of the evolution generated in Step 1, which correspond to this paper Fanti et al. and Zhu et al. respectively.

**Step 3**: Run the diagnosis algorithms proposed in this paper, by taking as input the corresponding observation, and record the running time at each step of the observation.

The generated evolution in Step 1 is shown in Figure 10. For this paper, the corresponding observation of the evolution is demonstrated in Figure 11, which is represented as $w_1 = M_0 e_1 M_1 e_2 M_2 \ldots e_3 M_{33}$. 
Meanwhile, for the approaches in Fanti et al.\textsuperscript{8} and Zhu et al.\textsuperscript{29} the observation of the evolution is
\[ \omega_2 = aabcbdgdabaaabcdddgdacddg, \]
that is, these two approaches have the same observation. Since the length of \( \omega_2 \) is 28 and the approaches in Fanti et al.\textsuperscript{8} and Zhu et al.\textsuperscript{29} use an online manner to perform diagnosis, we take \( \omega'_i = M_0e_1M_1 \cdots e_iM_i \) with \( i = 1, \ldots, 28 \) as an input of Algorithm 2 in turn.

The running times of these three algorithms when observing a step of the corresponding observation are shown in Tables 4 and 5, which are tested on a laptop with Intel Core i7-1165G7 processor and 16G RAM using Gurobi Solver (academic license).\textsuperscript{46} The relationship of running times of these three approaches in each step is visualized in Figure 12.

We observe that the running times of the approaches in Fanti et al.\textsuperscript{8} and Zhu et al.\textsuperscript{29} significantly increase as the increase of the length of the observation. The main reason for such a situation is that the size of the ILP problems built in Fanti et al.\textsuperscript{8} and Zhu et al.\textsuperscript{29} is linear with the length of an observation. However, the size of the ILP problem constructed in this paper is regardless of the length of an observation, as indicated by equations (6) and (7). Thus, the running time of our approach in each step remains basically the same and does not quickly increase when the observation becomes longer. In this case study, our approach is more efficient in contrast to the ones in Fanti et al.\textsuperscript{8} and Zhu et al.\textsuperscript{29}

### Table 4. Running times (in second) for steps 1–14.

| Step index | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14  |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----  |
| This paper | 0.6| 0.7| 0.5| 0.4| 0.6| 0.3| 0.3| 0.6| 0.4| 0.4| 0.7| 0.6| 0.7| 0.9  |
| Fanti et al.\textsuperscript{8} | 0.2| 0.3| 0.6| 0.4| 0.5| 0.8| 1.0| 1.1| 2.0| 2.3| 2.9| 5.1| 5.7| 7.7  |
| Zhu et al.\textsuperscript{29} | 0.5| 0.6| 0.2| 0.1| 0.2| 0.2| 0.3| 0.3| 0.4| 0.5| 0.7| 0.8| 1.1| 1.7  |

### Table 5. Running times (in second) for steps 15–28.

| Step index | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28  |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----  |
| This paper | 0.6| 0.7| 1.2| 0.9| 0.9| 1.4| 0.9| 0.8| 1.3| 1.1| 0.7| 0.8| 0.7| 0.8  |
| Fanti et al.\textsuperscript{8} | 14.1| 14.8| 20.0| 35.8| 39.2| 54.7| 96.3| 105.7| 148.6| 217.5| 215.1| 213.6| 422.2| 561.7 |
| Zhu et al.\textsuperscript{29} | 2.3| 2.4| 4.7| 4.6| 5.9| 9.9| 11.1| 15.7| 28.7| 40.0| 21.8| 44.7| 111.2| 154.4 |
**Conclusion**

In this paper, we solve the fault diagnosis problem using Petri net models. Two fault diagnosis algorithms for Petri nets and LPNs are developed, respectively. The proposed approach enjoys higher computational efficiency compared with those in Fanti et al.\(^8\) and Zhu et al.\(^{29}\) and is more suitable for use in real-world systems with low computing power. Future works are two-fold. Firstly, we plan to employ the proposed approach to a real-world system and report its effectiveness. Secondly, the proposed approach will be extended to stochastic Petri nets by associating a firing probability with each transition.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was funded in part by the National Natural Science Foundation of China under Grant No. 62103349, in part by the Science and Technology Development Fund, Macau SAR under Grant 0012/2019/A1, and in part by the Young and Middle-aged Scientific Research Basic Ability Promotion Project of Guangxi under Grant No. 2020KY02031.

**ORCID iD**

Guanghui Zhu [https://orcid.org/0000-0002-1627-2946](https://orcid.org/0000-0002-1627-2946)

**References**

1. Cassandras CG and Lafortune S. *Introduction to discrete event systems*. New York, NY: Springer, 2009.
2. Zad SH, Kwong RH and Wonham WM. Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Trans Automat Contr* 2003; 48(7): 1199–1212.
3. López-Estrada FR, Rotondo D and Valencia-Palomo G. A review of convex approaches for control, observation and safety of linear parameter varying and Takagi-Sugeno systems. *Processes* 2019; 7(11): 814.
4. Witczak M. Fault diagnosis and fault-tolerant control strategies for non-linear systems. New York, NY: Springer, 2014.
5. Blanke M, Kinnairt M, Lunze J, et al. *Diagnosis and fault-tolerant control*. New York, NY: Springer, 2006.
6. Sampath M, Sengupta R, Lafortune S, et al. Diagnosticity of discrete-event systems. *IEEE Trans Automat Contr* 1995; 40(9): 1555–1575.
7. Dotoli M, Fanti MP, Mangini AM, et al. On-line fault detection in discrete event systems by Petri nets and integer linear programming. *Automatica* 2009; 45(11): 2665–2672.
8. Fanti MP, Mangini AM and Ukovich W. Fault detection by labeled Petri nets in centralized and distributed approaches. *IEEE Trans Autom Sci Eng* 2013; 10(2): 392–404.
9. Cabasino MP, Giua A and Seatzu C. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica* 2010; 46(9): 1531–1539.
10. Zhu G, Li Z, Wu N, et al. Fault identification of discrete event systems modeled by Petri nets with unobservable transitions. *IEEE Trans Syst Man Cybern Syst* 2019; 49(2): 333–345.
11. Zhu G, Li Z and Wu N. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* 2018; 96: 201–212.
12. Ru Y and Hadjicostis CN. Fault diagnosis in discrete event systems modeled by partially observed Petri nets. *Discrete Event Dyn Syst* 2009; 19(4): 551–575.
13. Commault C, Dion JM and Yacoub Agha S. Structural analysis for the sensor location problem in fault detection and isolation. *Automatica* 2008; 44(8): 2074–2080.
14. Doostmohammadian M and Rabiee HR. On the observability and controllability of large-scale IOT networks: reducing number of unmatched nodes via link addition. *IEEE Control Syst Lett* 2021; 5(5): 1747–1752.
15. Shu X, Guo Y, Yang H, et al. Reliability study of motor controller in electric vehicle by the approach of fault tree analysis. *Eng Fail Anal* 2021; 121: 105165.
16. Vásquez JW, Pérez-Zuñiga G, Sotomayor-Moriano J, et al. New concept of safeprocess based on a fault detection methodology: Super alarms. *IFAC-PapersOnLine* 2019; 52(14): 231–236.
17. Foroozanfar M, Doustmohammadi A and Nikravesh S. Exponential stability of Petri net systems. In: *Proceedings of the 6th international conference on electrical engineering ICEENG*, Cairo, Egypt, May 2008, pp.1–9. New York: Springer.
18. Lutz-Ley A and Lopez-Mellado E. Stability analysis of discrete event systems modeled by Petri nets using unfoldings. *IEEE Trans Autom Sci Eng* 2018; 15(4): 1964–1971.
19. Sampath M, Sengupta R, Lafortune S, et al. Failure diagnosis using discrete-event models. *IEEE Trans Control Syst Technol* 1996; 4(2): 105–124.
20. Chen Y, Li Z, Barkaoui K, et al. Compact supervisory control of discrete event systems by Petri nets with data
inhibitor arcs. *IEEE Trans Syst Man Cybern Syst* 2017; 47(2): 364–379.

21. Demongodin I and Koussoulas NT. Differential Petri net models for industrial automation and supervisory control. *IEEE Trans Syst Man Cybern Part C* 2006; 36(4): 543–553.

22. Ma Z, Li Z and Giua A. Characterization of admissible marking sets in Petri nets with conflicts and synchronizations. *IEEE Trans Automat Contr* 2017; 62(3): 1329–1341.

23. Ammour R, Leclercq E, Sanlaville E, et al. State estimation of discrete event systems for RUL prediction issue. *Int J Prod Res* 2017; 55(23): 7040–7057.

24. Yin X. Verification of prognosability for labeled Petri nets. *IEEE Trans Automat Contr* 2018; 63(6): 1828–1834.

25. He Z, Li Z and Giua A. Cycle time optimization for deterministic timed weighted marked graphs under infinite server semantics. *IEEE Trans Automat Contr* 2018; 63(8): 2573–2580.

26. Tong Y, Li Z, Seatzu C, et al. Verification of state-based opacity using Petri nets. *IEEE Trans Automat Contr* 2017; 62(6): 2823–2837.

27. Basile F, Chiachio P and Coppola J. A novel model repair approach of timed discrete-event systems with anomalies. *IEEE Trans Autom Sci Eng* 2016; 13(4): 1541–1556.

28. Yin X and Lafortune S. On the decidability and complexity of diagnosability for labeled Petri nets. *IEEE Trans Automat Contr* 2018; 63(8): 2573–2580.

29. Zhu G, Feng L, Li Z, et al. An efficient fault diagnosis approach based on integer linear programming for labeled Petri nets. *IEEE Trans Automat Contr* 2021; 66(5): 2393–2398.

30. Basile F, Chiachio P and De Tommasi G. An efficient approach for online diagnosis of discrete event systems. *IEEE Trans Automat Contr* 2009; 54(4): 748–759.

31. Wang Y, Zhu G and Wu N. Fault diagnosis of backward conflict-free Petri nets by generalized markings. *IEEE Access* 2020; 8: 154871–154880.

32. Giua A and Seatzu C. Fault detection for discrete event systems using Petri nets with unobservable transitions. In: Proceedings of the 44th IEEE conference on decision and control, Seville, Spain, 15–15 December 2005, pp.6323–6328. New York, NY: IEEE.

33. Cabasino MP, Giua A, Pucci M, et al. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. *Control Eng Pract* 2011; 19(9): 989–1001.

34. Lefebvre D. On-line fault diagnosis with partially observed Petri nets. *IEEE Trans Automat Contr* 2014; 59(7): 1919–1924.

35. Ramirez-Trevino A, Ruiz-Beltran E, Rivera-Rangel I, et al. Online fault diagnosis of discrete event systems. A Petri net-based approach. *IEEE Trans Autom Sci Eng* 2007; 4(1): 31–39.

36. Genc S and Lafortune S. Distributed diagnosis of discrete-event systems using Petri nets. In: *Proceedings of international conference on application and theory of Petri nets*, Eindhoven, The Netherlands, June 2003, pp.316–336. New York: IEEE.

37. Pencole Y and Subias A. Diagnosability of event patterns in safe labeled time Petri nets: a model-checking approach. *IEEE Trans Autom Sci Eng* 2022; 19: 1151–1162.

38. Yin X, Chen J, Li Z, et al. Robust fault diagnosis of stochastic discrete event systems. *IEEE Trans Automat Contr* 2019; 64(10): 4237–4244.

39. Yu H, Ma Z, Li Z, et al. Diagnosability enforcement in labeled Petri nets using supervisory control. *Automatica* 2021; 131: 109776.

40. Cabral FG and Moreira MV. Synchronous diagnosis of discrete-event systems. *IEEE Trans Autom Sci Eng* 2020; 17(2): 921–932.

41. Veras MZM, Cabral FG and Moreira MV. Distributed synchronous diagnosis of discrete event systems modeled as automata. *Control Eng Pract* 2021; 115: 104892.

42. Al-Ajeli A and Parker D. Fault diagnosis in labelled Petri nets: a Fourier–Motzkin based approach. *Automatica* 2021; 132: 109831.

43. Lafortune S, Lin F and Hadjicostis CN. On the history of diagnosability and opacity in discrete event systems. *Annu Rev Control* 2018; 45: 257–266.

44. Giua A and Silva M. Petri nets and automatic control: a historical perspective. *Annu Rev Control* 2018; 45: 223–239.

45. Murata T. Petri nets: properties, analysis and applications. *Proc IEEE* 1989; 77(4): 541–580.

46. Gurobi Optimization. Gurobi Optimizer, http://www.gurobi.com/ (2021, accessed 25 October 2021).