Online Fault Diagnosis of Labeled Petri Nets Based on Reachability Graphs and Topological Sorting

YA WANG\(^1\), LI YIN\(^2\), AND GUANGHUI ZHU\(^2\), (Member, IEEE)

\(^1\)School of Information Engineering, Xuchang University, Xuchang 461000, China
\(^2\)Institute of Systems Engineering and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Taipa, Macau

Corresponding author: Li Yin (liyin@must.edu.mo)

This work was supported in part by the Science and Technology Development Fund, Macau SAR, under Grant 122/2017/A3 and Grant 0002/2019/APD, and in part by the Macau University of Science and Technology through the Faculty Research under Grant FRG-19-005-MISE.

ABSTRACT In this article, we propose an online fault diagnosis approach for labeled Petri nets. When observing an event, the reachability graph of the unobservable subnet (i.e., the net consisting of all places, unobservable transitions, and the attached arcs between them) is first constructed and then a topological sort (i.e., an ordered list of markings) of the reachability graph is obtained. According to the topological sort, each marking in the reachability graph is associated with a vector that contains diagnosis information to form a new graph, called a diagnosis graph. Based on the diagnosis graph constructed for each observed event, an online algorithm is developed to perform diagnosis. When the considered Petri net systems have a small amount of unobservable transitions, the proposed approach enjoys a high computational efficiency compared with the existing ones using integer linear programming.

INDEX TERMS Petri net, reachability graph, topological sort, discrete event system.

I. INTRODUCTION

In recent decades, due to the rapid development of technology, large and complex systems are emerging in large numbers, which can be viewed as discrete event systems (DES) [1] at a certain level of abstraction. Fault diagnosis aims to ensure the safe and stable operation of these systems by isolating and repairing faults as soon as possible. Petri nets, as a typical model of DES, have been widely used in performance optimization [2]–[6], big data [7], [8] and supervisory control [9]–[14] because of their distributed nature and well-developed structural analysis techniques. In this article, we address the issue of fault diagnosis using Petri nets.

In the context of DES, the problem of fault diagnosis is originally defined and solved based on automata [15], where faults are modeled as unobservable events and a diagnosis is constructed off-line to perform online diagnosis. A reader is referred to [16]–[18] for a detailed discussion of automaton-based supervisory control and fault diagnosis. Unlike an automaton model whose states are explicitly represented, a Petri net has a compact structure and its state set is implicitly characterized by a linear equation, called a state equation. Consequently, in the automatic control community, many methodologies based on Petri nets are reported, which do not require the explicit enumeration of all states [19]–[23].

In [24], the faults are modeled as unobservable transitions and are identified by solving an integer linear programming (ILP) problem built according to the given fault-free Petri net and an observed Petri net language. In this study, faulty transitions representing the possible faults are not included in the given fault-free Petri net. Due to the existing of faults, the observed language possibly cannot be generated by the fault-free net. The problem of fault diagnosis is solved by adding unobservable transitions (i.e., identified faults) into the fault-free net such that the resulting Petri net generates the observed language. The study [25] extends the work [24] by assuming that some additional places are observable, where the observed output is a marking-transition sequence. Due to the introduction of observable places, a better estimation on faults can be obtained in contrast to the approach in [24].
On the other hand, there is a deluge of diagnosis approaches that address the diagnosis issue by assuming that faulty transitions are explicitly included in the given Petri net model of a plant and fault diagnosis focuses on inferring the occurrences of faulty transitions based on an observed transition sequence. Basile et al. define the notion of g-markings (i.e., markings in which the number of tokens can be a negative integer) and propose an online diagnosis approach based on g-markings and integer linear programming.

In [27], a diagnosis approach based on integer linear programming is reported, where an ILP model is constructed on the basis of the observed transition sequence. The diagnosis results are derived by assigning different objective functions to the ILP model and solving it. This approach is extended to the case of labeled Petri nets in [28]. Cabasino et al. introduce the notion of basis markings and propose a diagnosis approach by constructing the basis reachability graph (BRG) of a net, which is a compact characterization of the reachability graph. This approach is applicable to bounded nets and is extended to labeled Petri nets in [30].

Ru and Hadjicostis deal with the diagnosis problem using partially observed Petri nets (POPNs). They prove the equivalence of POPNs and labeled Petri nets in the sense of fault diagnosis and solve the diagnosis problem by constructing the reachability graph of a Petri net. This approach does not require to solve ILP problems due to the construction of the reachability graph. However, the dependency issue raised when computing the diagnosis information associated with each marking in the reachability graph is not addressed. This article improves the approach from three aspects: (1) the dependency issue between markings in the reachability graph is solved by computing a topological sort of the reachability graph, (2) the advantages and applicability of reachability-graph-based approaches are discussed, and (3) an online diagnosis algorithm based on topological sorting is proposed.

To avoid enumerating the whole reachability set of a Petri net, multiple approaches above-mentioned [25]–[30] solve the fault diagnosis problem using ILP technique. The size of the ILP model constructed in [25]–[28] is closely related to the number of places and the length of an observation, which increases linearly as the length of the observation increases. Moreover, given an observation (i.e., an event sequence) of a labeled Petri net, there may exist a huge number of transition sequences whose firings generate the observation, which implies that a large number of ILP models require to be solved to perform diagnosis (see Section V for an example).

Enumerating the reachability set of a net is of exponential complexity and it may be computationally prohibitive for a large-size net. However, in this article, we only compute the reachability set of the unobservable subnet (i.e., the net consisting of all places, unobservable transitions, and attached arcs between them) of a Petri net. If the considered Petri net contains a small amount of unobservable transitions, the size of the unobservable subnet is also small and the proposed approach based on the enumeration of reachability set of the unobservable subnet is significantly efficient in contrast to the ILP-based ones (since the size of the ILP models and the number of times of solving these ILP models in ILP-based approaches [26]–[28] are possibly considerably large even if the number of unobservable transitions is small, as illustrated by the example shown in Section V).

More specifically, we assume that the unobservable subnet of a Petri net is acyclic and has no source transitions (i.e., transitions that have no input place) such that the reachability graph of the unobservable subnet is acyclic and has finite nodes. By associating each marking in the reachability graph of the unobservable subnet with a row vector containing diagnosis information (called a diagnosis vector), a new graph called a diagnosis graph is defined. When observing an event of a labeled Petri net, the corresponding diagnosis graph is first constructed and then diagnosis vectors are updated according to a topological sort of the graph. We propose an online diagnosis algorithm that performs diagnosis based on the diagnosis graph constructed at each step.

This article is organized as follows. Section II presents basic definitions used throughout the paper as well as some preliminary results. In Section III, the problem to be solved is formally defined. Then, in Section IV, we define the notion of diagnosis graph and present an online diagnosis algorithm. In Section V, we compare the computational efficiency of the proposed approach with an ILP-based one by considering a labeled Petri net example. Finally, the conclusion is drawn in Section VI.

II. PRELIMINARY

This section recalls basic concepts of Petri nets and some preliminary results. For more details on Petri nets, the reader is referred to [1], [32].

A. BASICS OF PETRI NETS

A Petri net is a four-tuple $N = (P, T, Pre, Post)$, where $P = \{p_1, \ldots, p_m\}$ is the set of places with cardinality $m$, $T = \{t_1, \ldots, t_n\}$ is the set of transitions with cardinality $n$, $Pre : P \times T \rightarrow \mathbb{N}^1$ and $Post : P \times T \rightarrow \mathbb{N}$ are the pre- and post-incidence matrices, respectively, which specify the arcs of the net. Pictorially, places and transitions are represented by circles and bars, respectively. The incidence matrix of a net is denoted by $C = Post - Pre$.

For a transition $t \in T$, its preset (the set of input places) is defined as $Pre(t) = \{p \in P \mid Post(p, t) > 0\}$, and its postset (the set of output places) is defined as $Post(t) = \{p \in P \mid Post(p, t) > 0\}$. Analogously, we define the preset and postset of a place $p \in P$ as $Pre(p) = \{t \in T \mid Post(p, t) > 0\}$ and $Post(p) = \{t \in T \mid Pre(p, t) > 0\}$, respectively. A transition $t \in T$ is called a source transition if it has no input place, i.e., $Pre(t) = \emptyset$.

A directed cycle of net $N$ is represented as $\tau_1 \ldots \tau_r$, where $\tau_i \in T$ for $i = 1 \ldots r$, $\tau_i \in \tau_{i-1}$ for $i = 2 \ldots r$, and $\tau_1 = \tau_r$.

A self-loop is the simplest form of a directed cycle.
cycle. A Petri net is said to be acyclic if it has no directed cycle.

A marking of a Petri net is a mapping \( M : P \rightarrow \mathbb{N} \) that assigns to each place a non-negative integer, denoted by \( M(p) \), representing the number of tokens in the place. Graphically, a token is usually denoted by a black dot. For economy of space, we write \( \alpha_1 p_1 + \alpha_2 p_2 + \cdots + \alpha_m p_m \) to represent a marking \( [\alpha_1 \alpha_2 \cdots \alpha_m]^T \). By associating an initial marking \( M_0 \) with a net \( N \), we obtain a Petri net system, denoted as \( \langle N, M_0 \rangle \).

A transition \( t \in T \) is said to be enabled at marking \( M \) if there are sufficient tokens in its input places, i.e., \( \forall p \in t^\bullet, M(p) \geq \text{Pre}(p, t) \). This situation is denoted by \( M[t] \).

A new marking \( M' \) is reached by firing an enabled transition \( t \) at \( M \), which is denoted as \( M[t]M' \). The reached marking \( M' \) can be computed by the state equation, i.e., \( M' = M + C(t, t) \).

A transition sequence \( \sigma = t_1 t_2 \cdots t_h \) with \( h \in \mathbb{N} \) is enabled at \( M \) if it holds \( M[t_1]M[t_2]M_2 \cdots [t_h]M_{P_h} \), which is denoted by \( M[\sigma] \). The final marking \( M_h \) is said to be reachable from \( M \) by firing \( \sigma \), denoted as \( M[\sigma]M_h \). We write \( t \in \sigma \) to represent that \( t \) is a transition contained in transition sequence \( \sigma \). The reachability set of Petri net system \( \langle N, M_0 \rangle \), denoted by \( R(N, M_0) \), is the set of all markings reachable from \( M_0 \). The language of Petri net \( N \) is the set of transition sequences that are enabled from \( M_0 \), formally defined by

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

Given a transition sequence \( \sigma \in T^* \), we define a function \( \pi : T^* \rightarrow \mathbb{N}^\sigma \) to compute the number of occurrences of transition \( t \in T \) in \( \sigma \) such that \( \gamma = \pi(\sigma) \) and \( \gamma(t) = h \) if \( t \) is contained \( h \) times in \( \sigma \). A marking \( M \) that is reachable from \( M_0 \) by firing a sequence \( \sigma \in T^* \), i.e., \( M_0[\sigma]M \), can be computed by the following state equation:

\[
M = M_0 + C \cdot \pi(\sigma),
\]
which provides a linear algebraic characterization of the state space of a net system.

A Petri net system \( \langle N, M_0 \rangle \) is said to be bounded if there exists a positive integer \( K \) such that \( M(p) \leq K \) for all \( p \in P \) and all \( M \in R(N, M_0) \). If a Petri net system is bounded for any initial marking \( M_0 \), then we say that it is structurally bounded.

Due to the absence of sensors, the firings of some transitions cannot be observed. Thus, the transition set is partitioned into two disjoint subsets \( T_o \) and \( T_u \) with \( T = T_o \cup T_u \), where \( T_o \) denotes the set of observable transitions with cardinality \( n_o \) and \( T_u \) is the set of unobservable transitions with cardinality \( n_u \).

In a Petri net for fault diagnosis, a fault is often characterized by an unobservable transition, called faulty transition (an observable fault can be readily detected). Meanwhile, there also exist some regular unobservable transitions. As a result, the unobservable transition set \( T_u \) is further divided into the set \( T_f \) of faulty transitions with \( n_f = |T_f|^2 \) and the set \( T_{\text{reg}} \) of regular unobservable transitions with \( n_{\text{reg}} = |T_{\text{reg}}| \).

\[\text{i.e., } T_u = T_f \cup T_{\text{reg}}. \] Obviously, it holds \( n = n_o + n_u = n_o + n_f + n_{\text{reg}}. \)

**Definition 1:** Given a subset of transitions \( T' \subseteq T \), the \( T' \)-induced subnetwork of a net \( N = (P, T, \text{Pre}, \text{Post}) \) is a net denoted as \( N' = (P, T', \text{Pre}', \text{Post}') \), where \( \text{Pre}' \) and \( \text{Post}' \) are the restrictions of \( \text{Pre} \) and \( \text{Post} \) to \( T \times T' \), respectively, i.e., the net \( N' \) consists of places in \( P \), transitions in \( T' \), and the associated arcs between them.

Given a net \( N = (P, T, \text{Pre}, \text{Post}) \) with \( T = T_o \cup T_u \), the \( T_o \)-induced subnetwork is called the unobservable subnet of \( N \), denoted by \( N_u = (P, T_u, \text{Pre}_u, \text{Post}_u) \). The incidence matrix of \( N_u \) is represented by \( C_u = \text{Post}_u - \text{Pre}_u \).

### B. LABELED PETRI NET

In a real-world system, because of financial or technical reasons, a sensor is usually used to monitor two or more system events whose occurrences have the same output. To model such a situation, we assign to each transition a label by defining a labeling function \( \ell : T \rightarrow E \cup \{\varepsilon\} \), where \( E \) is the given event set and \( \varepsilon \) stands for an empty string. An unobservable transition \( t \in T_u \) is associated with a label \( \varepsilon \) and thus we have \( T_u = \{ t \in T \mid \ell(t) = \varepsilon \} \). At the same time, we assign an observable transition \( t \in T_o \) a label from set \( E \) and have \( T_o = \{ t \in T \mid \ell(t) \in E \} \).

The set of transitions that are assigned a label \( \varepsilon \in E \) is denoted by \( T_\varepsilon = \{ t \in T \mid \ell(t) = \varepsilon \} \).

A labeled Petri net system \( \langle N, M_0, E, \ell \rangle \) is a Petri net system \( \langle N, M_0 \rangle \) that is associated with a labeling function \( \ell : T \rightarrow E \cup \{\varepsilon\} \). We point out that the class of Petri nets can be seen as a special subclass of labeled Petri nets by assuming \( E = T_o \) and assigning each observable transition \( t_0 \in T_o \) a unique label \( e \in E \) (i.e., \( \ell(t_0) = t_0 \in E \)).

The labeling function \( \ell \) can be extended to a sequence \( \sigma \in T^* \) such that (i) \( \ell(e) = e \) if \( \sigma = e \); (ii) \( \ell(\sigma) = \ell(t_1)\ell(t_2) \cdots \ell(t_h) \) if \( \sigma = t_1 t_2 \cdots t_h \) with \( h \) being a positive integer. Clearly, for a sequence \( \sigma \in T^* \), the sequence \( \omega = \ell(\sigma) \) consists of the labels of all observable transitions in \( \sigma \) and is called an observed word. Given a labeled net system \( \langle N, M_0, E, \ell \rangle \), we use

\[
L^E(N, M_0) = \{ \omega \in E^* \mid \exists \sigma \in L(N, M_0), \omega = \ell(\sigma) \}
\]
to denote the set of its observed words. Given an observed word \( \omega \in L^E(N, M_0) \), we define

\[
\Sigma(\omega) = \{ \sigma t_o \in T^* \mid \sigma t_o \in L(N, M_0), \ t_o \in T_o, \ \omega = \ell(\sigma t_o) \}
\]
as the set of consistent transition sequences with \( \omega \). In plain terms, the set \( \Sigma(\omega) \) contains all transition sequences \( \delta's \) in \( L(N, M_0) \) such that \( \omega = \ell(\delta) \) and the last transition of \( \delta \) is observable.

### C. TOPOLOGICAL SORTING

**Definition 2:** A directed graph is a two-tuple \( \mathcal{G} = (V, E) \), where \( V \) is the set of vertices and \( E \subseteq V \times V \) is the set of directed edges such that, for \( v_1, v_2 \in V, (v_1, v_2) \in E \) if there...
exists a directed arc from \( v_1 \) to \( v_2 \). An acyclic directed graph is a directed graph not containing directed cycles.

**Definition 3:** A topological sort of an acyclic directed graph \( G = (V, \mathcal{E}) \) is a linear ordering of \( V \) such that for each directed edge \((v_1, v_2) \in \mathcal{E}\), \( v_1 \) comes before \( v_2 \) in the linear ordering.

Originally, topological sorting is used to schedule a sequence of jobs based on their dependencies such that some jobs must be completed before a specific one. In this article, each marking in the reachability graph of the unobservable subnet is associated with a vector including diagnosis information, which is computed based on a topological sort of the reachability graph. One of algorithms for topological sorting is proposed by Kahn [33]. To make the paper self-contained, we show Kahn’s algorithm as follows.

**Algorithm 1:** Topological Sorting

**Input:** An acyclic directed graph

**Output:** A topological sort \( \mathcal{O} \)

1. Let \( \mathcal{O} \) be an empty linear list;
2. repeat
   3. Select a vertex \( \theta \) with no input arc;
   4. Add \( \theta \) to the tail of \( \mathcal{O} \);
   5. Delete \( \theta \) and all edges attached to \( \theta \) from \( G \);
3. until cannot find a vertex with no input arc;
4. Output a topological sort \( \mathcal{O} \);

According to Algorithm 1, the topological sort of an acyclic directed graph is not unique. For example, the reachability graph shown in Figure 1 is obviously an acyclic directed graph and its two topological sorts are \( \mathcal{O}_1 = \{M_0, M_1, M_2, M_3, M_4, M_5\} \) and \( \mathcal{O}_2 = \{M_0, M_1, M_3, M_2, M_4, M_5\} \), respectively.

**FIGURE 1.** An acyclic reachability graph.

**III. PROBLEM STATEMENT**

Fault diagnosis aims to detect and isolate fault events occurring in a system as soon as possible according to the output of the system. In the context of labeled Petri nets, fault events are usually modeled as faulty transitions (i.e., \( T_f \)) and the output is characterized in terms of event sequence, i.e., an observed word \( \omega \in \mathcal{E}^* \).

A faulty transition \( t_f \in T_f \) has three possible diagnosis decisions when observing an event \( e \in \mathcal{E} \), represented as \( N, U, \) and \( F \), meaning that \( t_f \) does not occur, \( t_f \) may have occurred, and \( t_f \) necessarily have occurred, respectively. The key point of fault diagnosis consists in inferring the diagnosis decision of \( t_f \) based on the output \( \omega \) of a Petri net system. Formally, the diagnosis result is described by a function, as defined as follows.

**Definition 4:** The diagnosis function \( \Phi : E^* \times T_f \to \{N, U, F\} \) of a labeled Petri net \( \langle N, M_0, E, \ell \rangle \) associates a fault \( t_f \in T_f \) with a diagnosis decision corresponding to the observed word \( \omega \in L^E(N, M_0) \) such that

1. \( \Phi(\omega, t_f) = N \) if for all \( \sigma \in \Sigma(\omega) \), \( t_f \notin \sigma \), i.e., each transition sequence consistent with \( \omega \) does not contain fault transition \( t_f \).
2. \( \Phi(\omega, t_f) = U \) if there exist \( \sigma_1, \sigma_2 \in \Sigma(\omega) \) such that \( t_f \notin \sigma_1 \) and \( t_f \in \sigma_2 \), i.e., one of the consistent sequences contains \( t_f \) and the other does not.
3. \( \Phi(\omega, t_f) = F \) if for all \( \sigma \in \Sigma(\omega) \), \( t_f \in \sigma \).

Before formally defining the diagnosis problem to be solved, we discuss the assumptions made in this article. The following assumptions are adopted:

(A1) The unobservable subnet is acyclic.

(A2) The unobservable subnet has no source transitions.

Assumption (A1) is usually made in the literature [24]–[30] for fault diagnosis to avoid the situation that a system runs in a cycle containing unobservable transitions only (no output is observed in such a case). We make Assumptions (A1) and (A2) to ensure that the reachability graph of the unobservable subnet is acyclic and has finite nodes such that its topological sort can be obtained.

**Proposition 1:** If a Petri net is acyclic and has no source transitions, then the Petri net is bounded and there is no directed cycle in its reachability graph.

**Proof:** The conclusion that the Petri net is bounded can be directly derived from Proposition 3 of [34]. We next prove, by contradiction, that there is no directed cycle in its reachability graph.

Suppose that the net is acyclic and there is a directed cycle in its reachability graph. If the cycle contains only one transition named \( t_1 \) (i.e., there exists a marking \( M \in R(N, M_0) \) such that \( M(t_1)M \)), a part of the net can be simplified as Figure 2(a1). The tokens “consuming” by the firing of \( t_1 \) must be recovered by \( t_1 \) (since the cycle contains only one transition). Thus, there must exist a directed cycle \( p_1t_1p_1 \) in the net that is illustrated by Figure 2(a2). This contradicts the assumption that the net is acyclic. If the cycle contains
two transitions, named \( t_1 \) and \( t_2 \) respectively, a part of the net can be simplified as Figure 2(b1). To avoid forming a self-loop, the tokens in \( p_1 \) are necessarily recovered by \( t_2 \) and the tokens in \( p_2 \) are recovered by \( t_1 \), which is illustrated by Figure 2(b2). Obviously, there is a directed cycle \( p_1 t_1 p_2 t_2 p_1 \) in the net, which contradicts the assumption. This contradiction is readily verified when the cycle contains more than two transitions and thus we prove the conclusion.

\[ S(M_0 \rightarrow M) = \{ \sigma \in T^* \mid M_0[\sigma]M \} \]

\[ S(M_0 \leftarrow M) = \{ \sigma \in T^* \mid M_0[\sigma]M, t \in \sigma \}. \]

**Definition 5:** Given a Petri net system \( \langle N, M_0 \rangle \), a diagnosis couple is a couple \((M, k)\), where \( M \in R(N, M_0) \) is a marking and \( k \in \mathbb{N}^{n_f+1} \) is an \((n_f+1)\) dimensional row vector, called a diagnosis vector, such that

- \( k(t_f) = |S(M_0 \rightarrow M)| \) for each \( t_f \in T_f \),
- \( k(n_f + 1) = |S(M_0 \leftarrow M)|. \)

Note that we write \( k(t_f) \) to denote the entry corresponding to fault transition \( t_f \) in \( k \), and \( k(n_f + 1) \) to denote the last entry of \( k \). In plain terms, the first \( n_f \) entries of \( k \) denote the number of paths from \( M_0 \) to \( M \) that pass the corresponding fault transition in the reachability graph and the entry \( k(n_f + 1) \) stands for the number of all paths from \( M_0 \) to \( M \).

**Definition 6:** Given an observed word \( \omega \in L^E(N, M_0) \) in a labeled Petri net \( \langle N, M_0, E, \ell \rangle \), the corresponding diagnosis group is a set of diagnosis couples, defined as

\[ \mathcal{X}(\omega) = \{ (M, k) \mid M \in R(N, M_0), k \in \mathbb{N}^{n_f+1}, \exists \sigma \in \Sigma(\omega), M_0[\sigma]M, \text{ and } k \text{ is defined according to Definition 5} \}. \]

The set \( \mathcal{X}(\omega) \) is not straightforward and is usually not trivial to obtain in a labeled Petri net. However, we show that it can be computed recursively in an acyclic net without source transitions by developing an algorithm (see Algorithm 2).

\[ k_0=[0,1], k_1=[0,1], k_2=[0,1], k_3=[1,1], \]
\[ k_6=[2,2], k_7=[2,2], k_8=[3,3], k_9=[1,1] \]
as the sum of all $k$'s in $\cal X(\omega)$. According to set $\cal X(\omega)$, the diagnosis result can be directly obtained, as shown in the following theorem.

**Theorem 1:** Given an observed word $\omega \in L^k(N, M_0)$ in a labeled Petri net $(N, M_0, E, \ell)$, for each $t_f \in T_f$, the following statements hold

1. $\Phi(\omega, t_f) = N$ if $\tilde{k}(t_f) = 0$,
2. $\Phi(\omega, t_f) = F$ if $\tilde{k}(t_f) = \tilde{k}(n_f + 1)$,
3. $\Phi(\omega, t_f) = U$ otherwise.

**Proof:** According to Definition 5, the last entry of $k$, i.e., $k(n_f + 1)$, in a diagnosis couple $(M, k)$ represents the number of transition sequences $\sigma$'s satisfying $M_0[\sigma]M$. Moreover, the set $\cal X(\omega)$ contains all diagnosis couples $(M, k)$'s whose markings $M$'s are reachable from $M_0$ by firing consistent transition sequences with $\omega$. Thus, $\tilde{k}(n_f + 1)$ stands for the number of transition sequences consistent with $\omega$ and $\tilde{k}(t_f)$ represents the number of consistent transition sequences passing $t_f$. The condition $\tilde{k}(t_f) = 0$ implies for all $\sigma \in \Sigma(\omega), t_f \notin \sigma$, and thus $\Phi(\omega, t_f) = N$ by Definition 4. The condition $\tilde{k}(t_f) = \tilde{k}(n_f + 1)$ means that the numbers of consistent transition sequences and consistent transition sequences passing $t_f$ are equal, i.e., for all $\sigma \in \Sigma(\omega), t_f \in \sigma$. Thus we have $\Phi(\omega, t_f) = F$ by Definition 4. Conclusion (3) is readily proved according to the similar reasoning.

To conveniently describe the proposed algorithm, we give the definition of diagnosis graph. By regarding a diagnosis couple $(M, k)$ as a node of a graph, a diagnosis graph is defined as follows.

**Definition 7:** A diagnosis graph is an acyclic directed graph $G = (V, E, W)$, where $V$ is the set of vertices, i.e., the set of diagnosis couple $(M, k)$'s, $E \subseteq V \times V$ is the set of edges such that $((M_1, k_1), (M_2, k_2)) \in E$ if there is a directed arc from vertex $(M_1, k_1)$ to $(M_2, k_2)$, and $W: V \times V \rightarrow T$ is a mapping that associates a transition $t \in T$ with an edge.

The successors of a vertex $v \in V$ in a diagnosis graph is defined as $suc(v) = \{v' \in V \mid (v, v') \in E\}$. The topological sort of a diagnosis graph $G$ is represented by $\cal T = toposort(G)$. The reader is referred to Subsection II-C for the algorithm of topological sorting (i.e., Algorithm 1).

**Example 2:** Figure 3(b) is an example of a diagnosis graph. We observe that $suc((M_1, k_1)) = [(M_2, k_2), (M_3, k_3)]$. One of the topological sorts of Figure 3(b) is $\cal T = [(M_0, k_0), (M_1, k_1), (M_3, k_3), (M_7, k_7), (M_2, k_2), (M_4, k_4), (M_6, k_6), (M_5, k_5)]$. Assuming $\omega = ab$, it holds $\cal X(ab) = [(M_3, k_3), (M_6, k_6), (M_7, k_7)]$ and thus $\tilde{k} = [6 \ 6]$. Since $\tilde{k}(t_3) = \tilde{k}(n_f + 1)$, we have $\Phi(\omega, t_3) = F$ by Theorem 1.

For a labeled Petri net satisfying Assumptions (A1) and (A2), when observing an event $e \in E$, a diagnosis graph is constructed by enumerating the transition sequences enabled in the unobservable subnet. Then, the diagnosis group $\cal X(\omega)$ is computed based on the diagnosis graph and the diagnosis results for each step are obtained by Theorem 1. The detailed diagnosis procedure is shown by Algorithm 2.

**Algorithm 2:** Online Diagnosis Using Diagnosis Graphs and Topological Sorting

**Input:** A labeled net system $(N, M_0, E, \ell)$ satisfying Assumptions (A1) and (A2)

**Output:** Diagnosis results for each observed event $e$

1. $\omega = e, \cal R = \emptyset$;
2. $V_0 = \emptyset$; $k_0 = \{[M_0, k_0]\}$;
3. Wait until a new event $e$ is observed;
4. $\omega = aw, X = \cal X, \cal E = \emptyset, W = \emptyset$;
5. while $\tilde{V} \neq \emptyset$ do
6. Remove an element from $\tilde{V}$, say $v = (M, k)$;
7. for each $t \in T_v$ do
8. if $M \geq Prec(\cdot, t)$ then
9. $M' = M + C(\cdot, t)$;
10. if $\exists \tilde{v}' = (M', k') \in \cal V$ satisfying $\tilde{M} = M'$ then
11. $E = E \cup \{(v, \tilde{v}')\}, W(v, \tilde{v}') = t$;
12. else
13. $v' = (M', k_0), \tilde{V} = \tilde{V} \cup \{v', \tilde{v}'\}, V = V \cup \{v'; \tilde{v}'\}$;
14. Build the diagnosis graph $G = (V, E, W)$;
15. Obtain its topological sort $\cal T = toposort(G)$;
16. for each $v \in \cal T$ from head to tail do
17. Find the vertex $v$ corresponding to $t$ in $G$, represented as $v = (M, k)$;
18. Obtain the successors $suc(v) \subseteq \cal T$ in $G$;
19. for each $v' = (M', k') \in suc(v)$ do
20. $w = W(v, v')$;
21. if $k' = v_0$ then
22. if $t \in T_v$ then /* Case I */
23. Update the vertex $(M', k')$ in $G$ by setting $k' = k + w$ and $k'(t) = k'(n_f + 1)$;
24. else /* Case II */
25. Update $(M', k')$ in $G$ by setting $k' = k$;
26. else
27. if $t \in T_v$ then /* Case III */
28. Update $(M', k')$ in $G$ by setting $k'(i) = k'(i) + k(t)$ for $i \in T_T \cup \{n_f + 1\}], t$ and $k'(t) = k'(n_f + 1);$
29. else /* Case IV */
30. Update $(M', k')$ in $G$ by setting $k' = k' + k$;
31. $X = \emptyset$;
32. for each vertex $v = (M, k)$ in $\cal X$ do
33. for each $t \in T_e$ do
34. if $M \geq Prec(\cdot, t)$ then
35. $\tilde{M} = M + C(\cdot, t)$;
36. if $\exists \tilde{v} = (M, k) \in X$ satisfying $\tilde{M} = \tilde{M}$ then
37. Update $(\tilde{M}, \tilde{k})$ by $\tilde{k} = \tilde{k} + k$;
38. else
39. $\tilde{v} = (\tilde{M}, k), X = X \cup \{\tilde{v}\}$;
40. $k = 0$;
41. for each $v = (M, k) \in X$ do
42. $\tilde{k} = \tilde{k} + k$;
43. for each $t \in T_f$ do
44. if $k(t_f) = 0$ then
45. $\cal R(t_f) = \Phi(\omega, t_f) = N$;
46. else if $\tilde{k}(t_f) = \tilde{k}(n_f + 1)$ then
47. $\cal R(t_f) = \Phi(\omega, t_f) = F$;
48. else
49. $\cal R(t_f) = \Phi(\omega, t_f) = U$;
50. Output the diagnosis results $\cal R$; Goto 3;
At the beginning of this algorithm, the diagnosis graph has only one node \((M_0, k_0)\) (see the variable \(\mathcal{X}\) in Line 2) and \(k_0\) is initialized as \([0^1 \times n, 1]\). Meanwhile, the default diagnosis vector attached to each marking is defined as a vector of zeros in Line 2, i.e., \(v_0 = 0^{1 \times (n_f + 1)}\). Since the unobservable subnet is acyclic and has no source transitions, the while loop in Line 5 is ensured to terminate. Note that the diagnosis decisions for \(n_f\) faulty transitions are represented by a row vector \(R\) such that, for \(T_f = \{f_1, f_2, \cdots , f_{n_f}\}\), \(R = [\Phi(\omega, f_1) \Phi(\omega, f_2) \cdots \Phi(\omega, f_{n_f})]\). In summary, the flow of Algorithm 2 can be broken into four steps:

**Step 1**: Build the diagnosis graph corresponding to the current observed event (Lines 5–14);

**Step 2**: Obtain the topological sort of the diagnosis graph (Line 15);

**Step 3**: Update \(k\) of each vertex \((M, k)\) in the diagnosis graph according to the topological sort (from Line 16 to 30);

**Step 4**: Build the diagnosis group \(\mathcal{X}\) and make a diagnosis (from Line 31 to 49).

We next explain these four steps in detail.

In **Step 1**, we construct the reachability graph of the unobservable subnet and associate each marking with a diagnosis vector of zeros, i.e., \(v_0\) (except the initial marking \(M_0\) that is associated with \(k_0\) to constitute the first vertex \((M_0, k_0)\) considered when updating the diagnosis vector in **Step 3**). In the following example, we visualize the diagnosis graph constructed in **Step 1**. In **Step 3**, we update each diagnosis vector of zeros in the diagnosis graph built in **Step 1**. However, diagnosis vectors necessarily are updated in a specific order (i.e., each diagnosis vector in a vertex should be updated before all its successors), otherwise they possibly do not satisfy Definition 5. A topological sort of the diagnosis graph built in **Step 1** is ideally suited to form such a specific order.

In **Step 3**, for each element \(\bar{v}\) from head to tail in a topological sort (see Line 16), the vertex \(v\) corresponding to \(\bar{v}\) in the diagnosis graph \(G\) possibly has been updated. Thus, we first find the corresponding vertex \(v = (M, k)\) in \(G\) in Line 17 and then update all successors of \(v\). When updating each successor \((M', k')\) of \(v\), there are four cases based on the original value \(k'\) and the transition \(t\) between \((M, k)\) and \((M', k')\):

- **Case I**: \(k' = v_0\) and \(t \in T_f\) (i.e., Line 22);
- **Case II**: \(k' = v_0\) and \(t \notin T_f\) (i.e., Line 24);
- **Case III**: \(k' \neq v_0\) and \(t \in T_f\) (i.e., Line 27);
- **Case IV**: \(k' \neq v_0\) and \(t \notin T_f\) (i.e., Line 29).

In Cases I and II, the condition \(k' = v_0\) means that the vertex \((M', k')\) has not been updated. If \(t \in T_f\), we update \(k'\) by setting \(k' = k + k'(t)\). If \(t \notin T_f\), the assignment \(k'(t) = k'(t) + 1\) means that from \(M_0\) to \(M'\), all paths pass the faulty transition \(t\) since the last transition in all paths is \(t \in T_f\). Otherwise, if \(t \notin T_f\), we simply update \(k'\) by setting \(k' = k\) according to Definition 5.

In Cases III and IV, the condition \(k' \neq v_0\) implies that the diagnosis vector \(k'\) has been updated and further updates will be performed. If \(t \notin T_f\) (i.e., Case IV), the update \(k' = k' + k\) (i.e., for a transition \(t \in T_f\), the update is \(k'(t) = k'(t) + k(t)\)) means that there are \(k(t)\) additional paths from \(M_0\) to \(M'\) that contain \(t_f\) and go through the marking \(M\) in \((M, k)\). If \(t \in T_f\) (i.e., Case III), we first set \(k' = k + k\) for all faulty transitions in \(T_f\) \(\setminus t\) and then update \(k(t)\) by \(k'(t) = k(t) + k(n_f + 1)\) means that there are \(k(n_f + 1)\) additional paths from \(M_0\) to \(M'\) that pass the marking \(M\) as well as the last faulty transition \(t\).

In **Step 4**, to obtain the diagnosis group \(\mathcal{X}\), we inspect each vertex \((M, k)\) of the diagnosis graph \(G\) to decide if an observable transition \(t\) labeled \(e\) (i.e., \(t \in T_e\)) is enabled at \(M\) (see Lines 32 and 33), where \(e\) is the event observed at the current step. If \(t\) is enabled at \(M\), a new diagnosis couple \((\bar{M}, k)\) with \(\bar{M} = M + C(e, t)\) is obtained and stored into \(\mathcal{X}\). Note that if the set \(\mathcal{X}\) has contained an element \((\bar{M}, \bar{k})\) with \(\bar{M} = M\), then the new diagnosis couple \((\bar{M}, k)\) will not be stored and the existing element \((\bar{M}, \bar{k})\) is updated by \(\bar{k} = \bar{k} + k\) in Line 37.

**FIGURE 4.** A Petri net whose unobservable subnet is acyclic and has no source transitions.

*Example 3:* Let us consider the net shown in Figure 4, where \(T_a = \{t_1, t_2, t_3, t_5\}\), \(T_f = \{t_3\}\), \(T_a = \{t_3\}\), \(T_b = \{t_6, t_7\}\), and \(M_0 = [1 1 0 1 0 0 0]^{T}\). The unobservable subnet is acyclic and has no source transitions. Suppose that the observed word \(\omega = ab\). Two diagnosis graphs are generated for two observed events (i.e., \(a\) and \(b\)) respectively by Algorithm 2, as shown in Figure 5.

Specifically, when \(w = a\), the corresponding diagnosis graph \(G_a\) is first constructed according to **Step 1** of Algorithm 2, as illustrated by Figure 5(a), where \(v_0\) is a diagnosis vector of zeros implying that the diagnosis vector in each vertex has not been updated. The topological sort of \(G_a\) is \(\bar{G}_a = \{(M_0, k_0), (M_4, v_0), (M_3, v_0), (M_1, v_0), (M_2, v_0), (M_6, v_0), (M_5, v_0), (M_7, v_0)\}\). Then, according to **Step 3**, \(G_a\) is updated and illustrated by the unobservable part of Figure 5(b), i.e., a subgraph consisting of edges labeled unobservable transitions (pictorially depicted by dashed lines) and the attached vertices. By inspecting each vertex of \(G_a\), we find that \(t_4\) with label \(a\) is enabled at vertices \((M_2, k_2), (M_3, k_3), (M_5, k_3), (M_7, k_7)\). Finally, according to **Step 4**, we obtain the diagnosis group \(\mathcal{X}(a) = \{s_{M_8, k_0}, (M_9, k_0), (M_{10}, k_{10}), (M_{11}, k_{11})\}\). By Theorem 1, it is readily verified that \(\Phi(a, t_3) = U\). For \(\omega = ab\), the corresponding diagnosis graph \(G_{ab}\) is shown in the unobservable part of Figure 5(c). In Figure 5(c), we observe that the corresponding
diagnosis group \( X(ab) = \{(M_{12}, k_{12}), (M_{13}, k_{13})\} \). Thus, it is easy to conclude \( \Phi(ab, t_3) = F \).

**Theorem 2:** Given a labeled Petri net system \((N, M_0, E, \ell)\) satisfying Assumptions (A1) and (A2), Algorithm 2 provides a correct solution to Problem 1.

**Proof:** We prove this theorem by induction.

**(basis step)** If the observed word \( \omega \) is an empty string (i.e., \( \omega = \varepsilon \)), the corresponding diagnosis graph has only one node \((M_0, k_0)\) with \( k_0 = 0 \). According to Theorem 1, the diagnosis decision for each faulty transition \( t_f \in T_f \) is \( \Phi(\omega, t_f) = N \). On the other hand, for \( \omega = \varepsilon \), the set of consistent transition sequences with \( \omega \) is empty, i.e., \( \sum(\omega) = \emptyset \). Thus, by Definition 4, we have \( \Phi(\omega, t_f) = N \) for each \( t_f \in T_f \). This coincides with the diagnosis result obtained by applying Theorem 1, which implies the correctness of Algorithm 2 for \( \omega = \varepsilon \).

**(inductive step)** Assuming that Algorithm 2 provides a correct solution to Problem 1 for an observed word \( \omega \in E^* \), we next prove that it holds for the observed word \( \omega \varepsilon \), where \( \varepsilon \) is an observed event after \( \omega \). When observing a new event \( e \) (see Line 3), the objective of Algorithm 2 is to compute the diagnosis group \( X(\omega e) \). To achieve the objective, the diagnosis graph \( G_e \) for the new event \( e \) is assigned to contain all elements in \( X(\omega) \) as the set of vertices at first (see \( V = X \) in Line 4). Assumptions (A1) and (A2) ensure that Step 1 of Algorithm 2 can terminate. After Step 1, the diagnosis graph \( G_e \) with default diagnosis vector \( v_0 \) in each vertex (except \((M_0, k_0)\)) is obtained, which is acyclic and has finite nodes. Thus, we can obtain a topological sort of \( G_e \) and update all diagnosis vectors in \( G_e \) according to Cases I, II, III and IV such that each vertex of \( G_e \) satisfies Definition 5. According to the definition of symbol \( \Sigma(\omega e) \) in Definition 4 (i.e., the last transition is observable), we need to check if an observable transition \( t \) labeled \( e \) is enabled at each vertex of \( G_e \). If \( t \) is enabled at a vertex \((M, k)\), the reached diagnosis couple \((M, k)\) with \( M = M + C(\cdot, t) \) is stored into \( \chi(\omega e) \) (see Line 39). After checking each vertex of \( G_e \), we obtain the diagnosis group \( X(\omega e) \) (see Lines 31–39). Then, by Theorem 1, the diagnosis decision \( \Phi(\omega e, t_f) \) of each faulty transition \( t_f \in T_f \) is trivially concluded.

**V. NUMERICAL EXAMPLE**

In this section, we discuss the computational efficiency of the proposed approach by means of a labeled Petri net shown in Figure 6, which is originally presented in [35] and is slightly modified by using different labeling function and initial marking herein. Obviously, the net shown in Figure 6 satisfies Assumptions (A1) and (A2). Moreover, we have \( M_0 = 3p_1 + 10p_0 + 10p_{13}, T_a = \{t_1, t_2, t_8, t_{10}, t_{16}\} \), and \( T_f = \{t_8, t_{10}\} \). The labeling function of the net is represented as \( T_a = \{t_3, t_6, t_{11}, t_{14}\}, T_b = \{t_4, t_7, t_{12}, t_{15}\}, T_c = \{t_9, t_{13}, t_{17}\} \), and \( T_d = \{t_5\} \).

One of typical ILP-based diagnosis approaches for labeled Petri nets is proposed in [28]. The main drawback of the approach is the possibility of solving considerably large number of ILP problems even if the size of the considered net is small. For example, if an observed word of the net shown in Figure 6 is an event sequence of length 30

\[ \omega = abcabddabccababcababcaabcabca, \]

then there exist 59966 possible observable transition sequences whose firings generate the observed word \( \omega \) (one of which is \( t_1t_4t_5t_7t_8t_4t_5t_7t_4t_5t_7t_4t_5t_14 t_1t_4t_5t_7t_4t_5t_14t_1t_4t_5t_7t_4t_5t_7t_4t_5t_14 \)). To perform diagnosis when observing the thirtieth event (i.e., ‘a’) of \( \omega \), 383981 ILP problems require to be solved according to the Diagnoser Algorithm shown in Figure 3 of [28]. In such a case, the main
TABLE 1. Comparison of two approaches for observed events indexed 1–12.

| index | 1  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-------|----|---|---|---|---|---|---|---|---|----|----|----|
| event | a  | b | c | a | b | d | a | b | a | b | a | b |
| time I (s) | 0.07 | 0.18 | 0.31 | 0.13 | 0.27 | 0.59 | 0.04 | 0.04 | 0.07 | 0.18 | 0.29 | 0.40 |
| time II (s) | 0.02 | 0.04 | 0.03 | 0.06 | 0.12 | 0.03 | 0.02 | 0.06 | 0.13 | 0.25 | 0.43 | 0.61 |
| RG size | 70 | 100 | 120 | 134 | 206 | 188 | 36 | 54 | 78 | 180 | 217 | 235 |
| ILP solving | 8 | 24 | 19 | 40 | 82 | 22 | 10 | 38 | 80 | 137 | 244 | 323 |

TABLE 2. Comparison of two approaches for observed events indexed 13–24.

| index | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|-------|----|---|---|---|---|---|---|---|----|----|----|----|
| event | a  | b | c | a | b | a | b | d | c | a | a |
| time I (s) | 0.28 | 0.15 | 0.20 | 0.39 | 0.26 | 0.35 | 0.64 | 0.84 | 1.14 | 1.17 | 0.92 | 1.19 |
| time II (s) | 0.88 | 1.51 | 2.39 | 3.52 | 6.17 | 8.78 | 14.51 | 19.76 | 14.96 | 35.90 | 66.84 | 79.02 |
| RG size | 170 | 159 | 202 | 259 | 259 | 317 | 538 | 618 | 740 | 682 | 726 | 840 |
| ILP solving | 458 | 721 | 1177 | 1651 | 2605 | 3875 | 5659 | 7933 | 5728 | 13186 | 21987 | 24707 |

FIGURE 6. A labeled Petri net for numerical example.

computational cost stems from the solutions of 383981 ILP problems, resulting in a low computational efficiency.

Different from the approaches using integer linear programming, the proposed approach only constructs the reachability graph (which necessarily has finite nodes by Proposition 1) of the unobservable subnet to perform diagnosis. In Figure 6, the numbers of tokens in the initial marking are not so big and the structure of the unobservable subnet is small. The reachability graph of the unobservable subnet has relatively small number of markings. Thus, for the labeled net in Figure 6, the proposed approach enjoys a high computational efficiency in contrast to the one in [28], as shown in the following comparisons.

For the observed word \( \omega \) above-mentioned, we perform fault diagnosis using the proposed approaches and the one in [28] and the comparisons are shown in Tables 1, 2, and 3. In these tables, the row “index” denotes the index of each event in \( \omega \). The rows “time I (s)”, “time II (s)”, “RG size” and “ILP solving” denote the running time of our approach (i.e., Algorithm 2), the running time of the approach in [28], the number of nodes in the reachability graph of the unobservable subnet, and the number of solving ILP problems in [28], respectively, for each observed event. The running time is tested on a computer with Intel i7-7700 processor and the ILP problems are solved using GUROBI solver.

We observe that these two approaches have similar computational performance in the first 12 observed events since the observed word is very short. However, the number of solving ILP problems in [28] increases very rapidly (see “ILP solving” row) as the length of the observed word increases but the number of markings in the constructed reachability graph for each observed event increases slowly (see “RG size” row). In such a case, the proposed approach is more efficient. Meanwhile, we point out that the proposed approach is not always more efficient than the one in [28]. For example, if the initial marking is \( M_0 = 30p_1 + 50p_9 + 50p_{13} \), the reachability graph of the unobservable subnet is significantly large and our approach is less efficient than the one in [28]. As far as we know, there does not exist a diagnosis approach that is most efficient for all cases in contrast to the other ones. The users are responsible to select a more suitable approach according to the concrete system settings. The proposed approach is more suitable to Petri net systems where the proportion of unobservable transitions is relatively small.

VI. CONCLUSION

The paper addresses the problem of fault diagnosis of labeled Petri nets. The main contributions are the application of topological sorting to update the diagnosis vectors and an online diagnosis algorithm based on diagnosis graphs. The proposed approach is more appropriate to Petri net systems with a small
number of unobservable transitions. In future works, we plan to extend this approach to the case of timed Petri nets.

REFERENCES

[1] C. G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems. New York, NY, USA: Springer, 2009.
[2] Z. He, Z. Li, and A. Giua, “Performance optimization for timed weighted marked graphs under infinite server semantics,” IEEE Trans. Autom. Control, vol. 63, no. 8, pp. 2573–2580, Aug. 2018.
[3] X. Zan, Z. P. Wu, C. Guo, and Z. H. Yu, “A Pareto-based genetic algorithm for multi-objective scheduling of automated manufacturing systems,” Adv. Mech. Eng., vol. 12, no. 1, pp. 1–15, 2020.
[4] Z. He, Z. Li, and A. Giua, “Optimization of deterministic timed weighted marked graphs,” IEEE Trans. Autom. Sci. Eng., vol. 14, no. 2, pp. 1084–1095, Apr. 2017.
[5] L. Yang, Z. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Wu, “On multiplexity-aware influence spread in social networks,” IEEE Access, vol. 8, pp. 106705–106713, 2020.
[6] J. Zhou, J. Wang, and J. Wang, “A simulation engine for stochastic timed Petri nets and application to emergency healthcare systems,” IEEEC/CAA J. Automatica Sinica, vol. 6, no. 4, pp. 969–980, Jul. 2019.
[7] D. A. Zaitsev, “Verification of complex systems with special edge conditions by infinite Petri nets,” Autom. Control Comput. Sci., vol. 47, no. 7, pp. 403–412, Dec. 2013.
[8] D. A. Zaitsev, V. G. Sarbe, and A. I. Sleptsov, “Synthesis of continuous-valued logic functions defined in tabular form,” Cybern. Syst. Anal., vol. 34, no. 2, pp. 190–195, Mar. 1998.
[9] L. Li, F. Basile, and Z. Li, “An approach to improve permissiveness of supervisors for GMECs in time Petri net systems,” IEEE Trans. Autom. Control, vol. 65, no. 1, pp. 237–251, Jan. 2020.
[10] L. Yang, Z. Li, and A. Giua, “Containment of rumor spread in complex social networks,” Inf. Sci., vol. 506, pp. 113–130, Jan. 2020.
[11] X. Li, E. S. Abouel El-Meligy, M. A. El-Sherbeeny, and Z. Li, “Adaptive consensus of two coupled heterogeneous systems with bidirectional actions,” IEEE Access, vol. 8, pp. 35832–35841, 2020.
[12] Y. Liu, K. Cai, and Z. Li, “On scalable supervisory control of multi-agent discrete-event systems,” Automatica, vol. 108, Oct. 2019, Art. no. 108460, doi: 10.1016/j.automatica.2019.06.012.
[13] X. Li, Z. Yu, Z. Li, and N. Wu, “Group consensus via pinning control for a class of heterogeneous multi-agent systems with input constraints,” Inf. Sci., vol. 542, pp. 247–262, Jan. 2021, doi: 10.1016/j.ins.2020.05.085.
[14] D. You, S. Wang, and C. Seatzu, “Supervisory control of a class of Petri nets with unobservable and uncontrollable transitions,” Inf. Sci., vol. 501, pp. 635–654, Oct. 2019.
[15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, “Diagnosability of discrete-event systems,” IEEE Trans. Autom. Control, vol. 40, no. 9, pp. 1555–1575, Sep. 1995.
[16] Q. Chen, L. Yin, N. Wu, M. A. El-Meligy, M. A. F. Sharaf, and Z. Li, “Diagnosability of vector discrete-event systems using predicates,” IEEE Access, vol. 7, pp. 147143–147155, 2019.
[17] Y. Hu, Z. Ma, and Z. Li, “Design of supervisors for active diagnosis in discrete event systems,” IEEE Trans. Autom. Control, early access, Jan. 28, 2020, doi: 10.1109/TAC.2020.2970011.
[18] D. Wang, X. Wang, and Z. Li, “Nonblocking supervisory control of state-tree structures with conditional-preemption matrices,” IEEE Trans. Ind. Informat., vol. 16, no. 6, pp. 3744–3756, Jun. 2020.
[19] S. Wang, W. Guo, X. Guo, X. Jiang, D. You, K. Barkaoui, and M. Zhou, “Computation of an emptiable minimal siphon in a subclass of Petri nets using mixed-integer programming,” IEEE/CAA J. Automatica Sinica, early access, Jun. 2, 2020, doi: 10.1109/JAS.2020.1003210.
[20] G. Zhu, L. Feng, Z. Li, and N. Wu, “An efficient fault diagnosis approach based on integer linear programming for labeled Petri nets,” IEEE Trans. Autom. Control, early access, Jul. 13, 2020, doi: 10.1109/TAC.2020.3008712.
[21] D. You, S. Wang, and C. Seatzu, “Verification of fault-predictability in labeled Petri nets using predictor graphs,” IEEE Trans. Autom. Control, vol. 64, no. 10, pp. 4353–4360, Oct. 2019.
[22] N. Ran, H. Su, and S. Wang, “An improved approach to test diagnosability of bounded Petri nets,” IEEE/CAA J. Automatica Sinica, vol. 4, no. 2, pp. 291–303, Apr. 2017.
[23] D. Sun, Y. Chen, M. A. El-Meligy, M. A. F. Sharaf, N. Wu, and Z. Li, “On algebraic identification of critical states for deadlock control in automated manufacturing systems modeled with Petri nets,” IEEE Access, vol. 7, pp. 121332–121349, 2019.
[24] M. P. Cabasino, A. Giua, C. N. Hadjicostis, and C. Seatzu, “Fault model identification and synthesis in Petri nets,” Discrete Event Dyn. Syst., vol. 25, no. 3, pp. 419–440, Sep. 2015.
[25] G. Zhu, Z. Li, and N. Wu, “Model-based fault identification of discrete event systems using partially observed Petri nets,” Automatica, vol. 96, pp. 201–212, Oct. 2018.
[26] F. Basile, P. Chiacchio, and G. De Tommasi, “An efficient approach for online diagnosis of discrete event systems,” IEEE Trans. Autom. Control, vol. 54, no. 4, pp. 748–759, Apr. 2009.
[27] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Uckovich, “On-line fault detection in discrete event systems by Petri nets and integer linear programming,” Automatica, vol. 45, no. 11, pp. 2665–2672, Nov. 2009.
[28] M. P. Fanti, A. M. Mangini, and W. Uckovich, “Fault detection by labeled Petri nets in centralized and distributed approaches,” IEEE Trans. Autom. Sci. Eng., vol. 10, no. 2, pp. 392–404, Apr. 2013.
[29] M. P. Cabasino, A. Giua, and C. Seatzu, “Fault detection for discrete event systems using Petri nets with unobservable transitions,” Automatica, vol. 46, no. 9, pp. 1531–1539, Sep. 2010.
[30] M. P. Cabasino, A. Giua, M. Pocci, and C. Seatzu, “Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems,” Control Eng. Pract., vol. 19, no. 9, pp. 989–1001, Sep. 2011.
[31] Y. Ru and C. N. Hadjicostis, “Fault diagnosis in discrete event systems modeled by partially observed Petri nets,” Discrete Event Dyn. Syst., vol. 25, no. 4, pp. 561–575, Dec. 2009.
[32] T. Murata, “Petri nets: Properties, analysis and applications,” Proc. IEEE, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[33] A. B. Kahn, “Topological sorting of large networks,” Commun. ACM, vol. 5, no. 11, pp. 588–592, Nov. 1962.
[34] Y. Ru and C. N. Hadjicostis, “Bounds on the number of markings consistent with label observations in Petri nets,” IEEE Trans. Autom. Sci. Eng., vol. 6, no. 2, pp. 334–344, Apr. 2009.
[35] S. Gene and S. Lafortune, “Distributed diagnosis of discrete-event systems using Petri nets,” in Proc. Int. Conf. Appl. Theory Petri Nets, Eindhoven, The Netherlands, 2003, pp. 316–336.