mathematics

MDPI

Article

Polynomial-Time Verification of Decentralized Fault Pattern Diagnosability for Discrete-Event Systems

Ye Liang [1], Gaiyun Liu [1,*] and Ahmed M. El-Sherbeeny [2]

[1] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China; liangye@stu.xidian.edu.cn
[2] Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia; aelsherbeeny@ksu.edu.sa
* Correspondence: gyliu@xidian.edu.cn; Tel.: +86-29-8820-1986

Abstract: This paper considers the verification of decentralized fault pattern diagnosability for discrete event systems, where the pattern is modeled as a finite automaton whose accepted language is the objective to be diagnosed. We introduce a notion of codiagnosability to formalize the decentralized fault pattern diagnosability, which requires the pattern to be detected by one of the external local observers within a bounded delay. To this end, a structure, namely a verifier, is proposed to verify the codiagnosability of the system and the fault pattern. By studying an indeterminate cycle of the verifier, sufficient and necessary conditions are provided to test the codiagnosability. It is shown that the proposed method requires polynomial time at most. In addition, we present an approach to extend the proposed verifier structure so that it can be applied to centralized cases.

Keywords: discrete event system; decentralized diagnosis; fault pattern; codiagnosability; computational complexity

MSC: 93C65

Citation: Liang, Y.; Liu, G.; El-Sherbeeny, A.M. Polynomial-Time Verification of Decentralized Fault Pattern Diagnosability for Discrete-Event Systems. *Mathematics* **2023**, *11*, 3998. https://doi.org/10.3390/math11183998

Academic Editor: António Lopes

Received: 21 July 2023
Revised: 13 September 2023
Accepted: 18 September 2023
Published: 20 September 2023

1. Introduction

A discrete-event system (DES) [1–3] is an event-driven system whose state space is a discrete set and transitions from one state to another are triggered by the occurrence of events. The state of such a system may have logical or symbolic values that change in response to the event. In the context of DESs [2], fault diagnosis involves determining whether or not a fault has occurred with certainty [4,5]. The problem of diagnosis has been extensively studied [6–10]. Its formal description is located in [7]. The purpose of diagnosability is to determine whether any predetermined failure can be diagnosed within a finite delay (steps) after its occurrence. Experience with monitoring dynamic systems shows that there is a large spectrum of faulty situations in practical systems [11], such as multiple faults, intermittent faults [12], and temporary faults [13] that are not consistent with single-event faults. In order to consider these complex situations, such as temporary and/or intermittent faults, fault pattern diagnosis simultaneously emerges as a promising research area, which provides a general way to solve the diagnosis problems by capturing the occurrences of particular strings in a system and gains extensive attention from both researchers and practitioners, leading to a bulk of documented results [14–20].

As known, Petri nets (PNs) and finite state automata are two major tools to treat various problems in DESs. Specifically, finite state automata have been investigated for that purpose. In [14], the authors provide the ground foundations for the fault pattern diagnosis that relies on the synchronous product of a system with the pattern and the computation of the determinization of the resulting structure. Then, in [15], the same authors improve the complexity of the approach proposed in [14] by successively using the unobservable closure and a pair verifier to replace the determinization. This also introduces fault patterns in the problem of prediction. The authors of [16] address the fault pattern diagnosis by

means of state isolation. In contrast to the approach in [14], which builds a diagnoser with potentially exponential complexity, the method proposed in [16] performs state estimation in a recursive manner. This approach offers the advantage of lower complexity for fault pattern detection. In [17], the authors introduce different notions of diagnosability, namely S-type and T-type pattern diagnosability, depending on whether interweaving events are accepted in the pattern. Note that this work also focuses on S-type patterns (but the T-type patterns are viewed as a particular subclass of S-type patterns, as mentioned in [17]). Moreover, in comparison to existing works, the main contribution of this paper is not only on the centralized systems but also on the decentralized frameworks, which expands the use of fault patterns.

Some contributions also consider the fault pattern diagnosis issues in PNs. On the one hand, the authors in [21] work on the diagnosability analysis of PNs by transforming the diagnosis problem into model checking. Using a particular class of PNs, the diagnosis of fault patterns has been studied according to the matching operator in [22]. With a similar approach, the same authors extended their results to the diagnosability analysis in [23]. On the other hand, stochastic PNs, endowed with Markovian semantics, have been studied to extract weak diagnosability properties for fault patterns [24]. A few researchers also consider fault patterns from the time aspects [25,26].

For the purpose of diagnosability verification, a systematic structure, namely a diagnoser, is proposed in [7], which provides the necessary and sufficient conditions for diagnosability offline. It also can be used for online fault detection based on the observations of system behaviors. The state space of a diagnoser is in the worst case exponential with respect to the size of a system model. To overcome the potential state explosion problem, a so-called "twin machine" technique [9,10], is introduced to provide a worst-case polynomial test with respect to the number of states of a system for diagnosability, without constructing a diagnoser. A state-based method for DES diagnosability is proposed in [6], which provides an algorithm for computing a sequence of test commands to detect faults.

All aforementioned works are centralized fault diagnosis, in which a centralized diagnoser is responsible for the diagnosis in a system. However, many large-sized and complex systems are physically decentralized, where diagnosis information collected by decentralized local sites will be sent to a centralized site for analysis. In this case, a variable communication delay needs to be introduced, which makes the diagnosis technique more complicated. Although all diagnosis information can be collected centrally, due to the data delay, a centralized fault diagnosis method may not always be suitable for decentralized systems.

This work focuses on the decentralized diagnosis of the DESs with multiple local observers, each of which possesses its own sensor without involving any communication among diagnosers, so as to make up for the defects of the centralized diagnosis. The authors in [27] propose a coordinated decentralized architecture with a coordinator, which extends the notion of diagnosability from centralized systems [7] to decentralized cases. In [28], a failure is characterized as a violation of a specification represented by a language, and codiagnosability is studied in a specification language framework, which is later specialized to the failure event framework. In [29], a hierarchical paradigm is developed by incorporating the protocol in [28], which leads to a hierarchy of architectures and encompasses many existing structures for decentralized diagnosis. Considering global consistency conditions, the authors in [30] discuss the diagnosis problem of the distributed setting with fault pattern and define local pattern recognizers for that purpose.

This paper deals with the decentralized fault pattern diagnosability verification of DESs, where the fault patterns are modeled by finite automata. We develop a codiagnosability notion to formalize the decentralized pattern diagnosability of the system and provide an algorithm for its verification. We first compute a synchronous product structure, which encodes a system and a pattern. A failure diagnoser structure is constructed based on the synchronous product by considering the faulty behaviors of the system, and a non-failure diagnoser is calculated based on the normal behaviors. Then, we present a local non-failure

diagnoser based on the local projection. In order to verify the codiagnosability of the system and the pattern, a verifier structure is derived by taking the product of failure diagnoser and local non-failure diagnoser. By studying the indeterminate cycles of the verifier, we establish necessary and sufficient verification conditions of codiagnosability. The successive steps of the proposed approach are visualized in Figure 1.

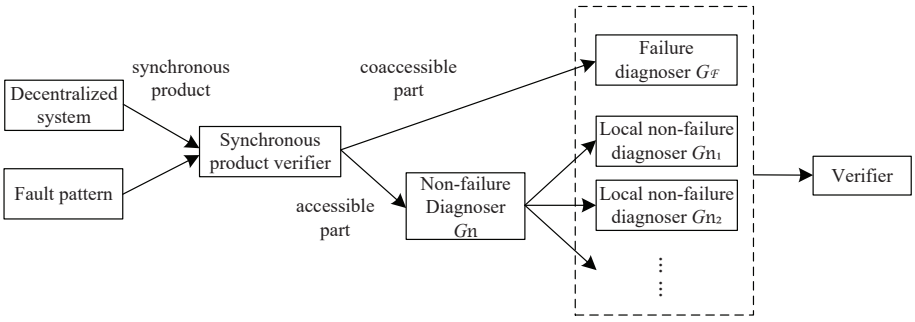

Figure 1. Methodology.

The first contribution of the paper focuses on the construction of failure and non-failure diagnosers, which track the normal and faulty behaviors of the system, separately. The second contribution is the computation of the local diagnoser with respect to the local projection. The third contribution is to extend the necessary and sufficient conditions stated for pattern diagnosability [14] to a decentralized setting. This is achieved by constructing a verifier structure, which is the product of the failure and local non-failure diagnosers. The proposed method is shown to be of polynomial complexity. We also present a method to extend the proposed verifier so that it can be applied to centralized cases.

The rest of the paper is organized as follows. Section 2 reviews finite state automata. Section 3 begins with the notions of fault patterns and then touches upon decentralized fault pattern diagnosis in DESs. Section 4 provides an algorithm for fault pattern codiagnosability with the local diagnosers. Section 5 analyses the complexity of the proposed method. Section 6 concludes this research.

2. Preliminaries

In this paper, we use $\mathbb{N}$ to denote the set of strictly positive integers. In what follows, we consider the model of finite automaton and its related notions.

Definition 1. *A deterministic finite automaton (DFA) is a four-tuple $G = (L, \Sigma, \delta, l_0)$, where L is the set of states, Σ is the set of events, l_0 is the initial state, and $\delta : L \times \Sigma \to L$ is the partial transition function: $l' = \delta(l, \sigma)$ means that there is a transition labeled with event $\sigma \in \Sigma$ from state l to state l'. Let Σ^* be the set of all finite strings defined over Σ, including the empty string λ. Transition function δ can be extended to $L \times \Sigma^* \to L$ in an usual way: given $l \in L$, $w \in \Sigma^*$, and $\sigma \in \Sigma$, $\delta(l, \lambda) = l$ and $\delta(l, w\sigma) = \delta((\delta(l, w)), \sigma)$.*

For a DFA G with partial observation, the events set Σ can be partitioned into two disjoint subsets $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o and Σ_{uo} represent the set of observable events and the set of unobservable events, respectively. Given two strings $w', w'' \in \Sigma^*$, the concatenation of the two strings is a string $w = w'w'' \in \Sigma^*$, where the string w' is followed by w''.

For a state $l \in L$, the set of active events at l is defined as $\Lambda(l) = \{\sigma \in \Sigma | \exists l' \in L : l' = \delta(l, \sigma)\}$. Given a string $w \in \Sigma^*$, its length is defined as the number of items in w, denoted by $|w|$. A string $w' \in \Sigma^*$ is said to be a prefix of $w \in \Sigma^*$ if there exists $w'' \in \Sigma^*$, such that $w = w'w''$. The language generated by the DFA G is defined as $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(l_0, s)!\}$, where $\delta(l_0, s)!$ means that "$\delta(l_0, s)$ is defined". Given a string $w \in \mathcal{L}(G)$, $\mathcal{L}(G)/w$ denotes

the post-language of $\mathcal{L}(G)$ after w, defined as $\mathcal{L}(G)/w = \{w' \in \Sigma^* \mid ww' \in \mathcal{L}(G)\}$. A run that begins with the initial state l_0 has the form:

$$\rho = l_0 \xrightarrow{\sigma_0} l_1 \xrightarrow{\sigma_1} \dots l_n \xrightarrow{\sigma_n} l_{n+1}, \tag{1}$$

where $l_i, l_{i+1} \in L, \sigma_i \in \Sigma$, and $l_{i+1} = \delta(l_i, \sigma_i)$ for $i \in \{0, 1, \dots, n\}$. In this case, we say that the run ρ is associated with the string $w = \sigma_0 \dots \sigma_n \in \Sigma^*$. A run is a cycle if $l_{n+1} = l_0$.

For a decentralized system G, we use Σ_{o_i} to denote the set of events observed by the local observer, simply called the local observable event set, and use Σ_{uo_i} to denote the set of local unobservable event set, where $\Sigma_{uo_i} = \Sigma \setminus \Sigma_{o_i}$, $i = 1, \dots, m$, $m \in \mathbb{N}$, m is the number of the local sites. For a set of observable events Σ_o of G, $\Sigma_o = \bigcup_{i=1}^{m} \Sigma_{o_i}$, $i = 1, \dots, m$. Without loss of generality, we assume that there is no cycle composed only of silent events in the system G. Note that in the decentralized architecture, different sites may have events in common, i.e., for all $i, j \in \{1, \dots, m\}$ and $i \neq j$, the set of $\Sigma_{o_i} \cap \Sigma_{o_j}$ is not necessarily an empty set. Given a site i, $i = 1, \dots, m$, the local projection $\mathscr{P}_i : \Sigma^* \to \Sigma_{o_i}^*$ is defined as follows: for $w \in \Sigma^*$ and $\sigma \in \Sigma$,

$$\mathscr{P}_i(w\sigma) = \begin{cases} \mathscr{P}_i(w)\sigma & \text{if } \sigma \in \Sigma_{o_i} \\ \mathscr{P}_i(w) & \text{otherwise,} \end{cases} \tag{2}$$

and $\mathscr{P}_i(\lambda) = \lambda$. In other words, a local projection function $\mathscr{P}_i$ tracks only its corresponding local observable events. For $i = 1, \dots, m$, a site i has its own set of observable events and does not communicate with each other. For a string $w \in \Sigma_{o_i}^*$, the inverse of the local projection $\mathscr{P}_i^{-1} : 2^{\Sigma_{o_i}^*} \to 2^{\Sigma^*}$ is defined by $\mathscr{P}_i^{-1}(\{w\}) = \{w' \in \mathcal{L}(G) |\ \mathscr{P}_i(w') = w\}$. Let $G_1 = (L_1, \Sigma_1, \delta_1, l_0^1)$ and $G_2 = (L_2, \Sigma_2, \delta_2, l_0^2)$. The parallel composition is defined as $G_1 || G_2 = \mathscr{P}_1^{-1}(G_1) \times \mathscr{P}_2^{-1}(G_2)$, where $\mathscr{P}_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$, $i = 1, 2$.

Example 1. *Consider a DFA G_1 shown in Figure 2 with respect to the local projection $\mathscr{P}_i$, $i = 1, 2$, where $L = \{0, 1, 2, 3, 4, 5, 6, 7\}$ is the set of states, 0 is the initial state, $\Sigma = \{a, b, c, u, f_1, f_2\}$ is the set of events, and $\Sigma_o = \{a, b, c\}$ is the set of observable events with $\Sigma_{o_1} = \{a, b\}$ and $\Sigma_{o_2} = \{a, c\}$. The transition function is defined as $\delta(0, a) = 1$, $\delta(1, f_2) = 2$, $\delta(2, b) = 7$, $\delta(2, c) = 7$, $\delta(7, a) = 7$, $\delta(7, c) = 7$, $\delta(2, f_2) = 3$, $\delta(3, b) = 4$, $\delta(4, c) = 5$, $\delta(5, a) = 6$, and $\delta(6, u) = 6$. The local projection $\mathscr{P}_1$ of system G_1 is defined as $\mathscr{P}_1(a) = a$, $\mathscr{P}_1(b) = b$, $\mathscr{P}_1(c) = \mathscr{P}_1(f_1) = \mathscr{P}_1(f_2) = \mathscr{P}_1(u) = \lambda$, and $\mathscr{P}_2$ is defined as $\mathscr{P}_2(a) = a$, $\mathscr{P}_2(c) = c$, and $\mathscr{P}_2(b) = \mathscr{P}_2(f_1) = \mathscr{P}_2(f_2) = \mathscr{P}_2(u) = \lambda$. A possible run generated by system G_1 from the initial state is*

$$\rho : 0 \xrightarrow{a} 1 \xrightarrow{f_1} 2 \xrightarrow{f_2} 3 \xrightarrow{b} 4 \xrightarrow{c} 5 \xrightarrow{a} 6,$$

where the associated string of ρ is $w = af_1f_2bca$. The local projection $\mathscr{P}_1$ of w with respect to Σ_{o_1} is $\mathscr{P}_1(w) = aba$, and $\mathscr{P}_2$ with respect to Σ_{o_2} is $\mathscr{P}_2(w) = aca$.

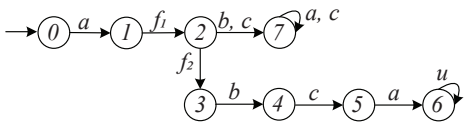

Figure 2. Deterministic finite automaton G_1.

3. Decentralized Fault Pattern Diagnosis of Automata

A fault pattern, simply called a pattern, is defined as a finite-state automaton whose accepted language is the objective to be diagnosed, which represents the occurrence of complex or composite faults.

Definition 2. *A (fault) pattern of a DFA $G = (L, \Sigma, \delta, l_0)$ is $\Omega = (S, \Sigma, \delta_\Omega, s_0, s_\Omega)$, where S is the set of states, Σ is the set of events, $s_0 \in S$ is the initial state, $s_\Omega \in S$ is the single final state, and*

$\delta_\Omega : S \times \Sigma \to S$ *is the transition function. The pattern* Ω *satisfies a complete condition, i.e., for all* $s \in S$, $\Lambda(s) = \Sigma$ *and the final state* s_Ω *is stable, i.e., for all* $\sigma \in \Sigma$, $\delta_\Omega(s_\Omega, \sigma) = s_\Omega$.

We use Σ_F to denote the set of target events that lead to the occurrence of the pattern, where $\Sigma_F \subseteq \Sigma_{uo}$. The language of pattern Ω, denoted by $\mathcal{L}(\Omega)$, satisfies $\mathcal{L}(\Omega) = \Sigma^*$ due to its complete condition. We use $\mathcal{L}_\mathcal{A}(\Omega)$ to denote the accepted language of Ω, defined as $\mathcal{L}_\mathcal{A}(\Omega) = \{w \in \mathcal{L}(\Omega) | \delta_\Omega(s_0, w) = s_\Omega\}$, and define the target language of DFA G as $\mathcal{L}_\mathcal{A}(G) = \mathcal{L}(G) \cap \mathcal{L}_\mathcal{A}(\Omega)$.

Example 2. *An example of pattern* Ω *is shown in Figure 3 with the set of states* $S = \{N_1, N_2, F\}$, *the set of target events* $\Sigma_F = \{f_1, f_2\}$, *the final state* $s_\Omega = F$, *and the initial state* $s_0 = N_1$. *The accepted language of the pattern* Ω *is* $\mathcal{L}_\mathcal{A}(\Omega) = \Sigma^* f_1 \Sigma^* f_2 \Sigma^*$.

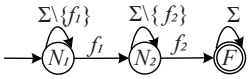

Figure 3. Pattern Ω.

In the following, we provide a formal definition, namely codiagnosability, with respect to the decentralized system and the pattern, detailed in Definition 3.

Definition 3. *Given a DFA* G, *a pattern* Ω, *and the local projection* $\mathcal{P}_i$, $i \in \{1, \ldots, m\}$, G *is codiagnosable with regard to* Ω *and* $\mathcal{P}_i$ *if*

$$(\exists k \in \mathbb{N})\ (\forall w \in \mathcal{L}_\mathcal{A}(G))\ (\forall w' \in \mathcal{L}(G)/w)$$
$$(|w'| \geq k) \Rightarrow (\exists i \in \{1, \ldots, m\})[\mathcal{P}_i^{-1}(\mathcal{P}_i(ww')) \subseteq \mathcal{L}_\mathcal{A}(G)].$$

According to Definition 3, the system G is codiagnosable with respect to the local projection $\mathcal{P}_i$ and the pattern Ω if and only if for any string ww' accepted by the pattern Ω, there does not exist a string w'', which is not accepted by Ω, such that $\mathcal{P}_i(ww') = \mathcal{P}_i(w'')$. An algorithm of fault pattern codiagnosability test is proposed based on searching for the strings $ww' \in \mathcal{L}_\mathcal{A}(G)$ and $w'' \notin \mathcal{L}_\mathcal{A}(G)$, such that, for $i = 1, \ldots, m$, the two strings ww' and w'' violate the codiagnosability condition of Definition 3.

4. Verification Algorithm

For the purpose of codiagnosability verification, we propose an algorithm and a theorem in this section. Definition 4 introduces a structure that will be used later, namely *synchronous product*, which encodes the system G and the pattern Ω.

Definition 4. *Given a DFA* $G = (L, \Sigma, \delta, l_0)$, *and a pattern* $\Omega = (S, \Sigma, \delta_\Omega, s_0, s_\Omega)$, *a synchronous product* G_Ω *of* G *with respect to* Ω *is a DFA* $G_\Omega = (L_{G_\Omega}, \Sigma, \delta_{G_\Omega}, l_0^{G_\Omega}, L_F^{G_\Omega})$, *where* $L_{G_\Omega} \subseteq L \times S$ *is the set of states,* Σ *is the set of events,* $l_0^{G_\Omega} = (l_0, s_0)$ *is the initial state,* $L_F^{G_\Omega} = L \times \{s_\Omega\}$ *is the set of final states, and* $\delta_{G_\Omega} : (L \times S) \times \Sigma \to (L \times S)$ *is the transition function, defined as follows: for* $\sigma \in \Sigma$, $s, s' \in S$, $l, l' \in L$, $\delta_{G_\Omega}((l, s), \sigma) = (l', s')$ *if* $\delta(l, \sigma) = l'$ *and* $\delta_\Omega(s, \sigma) = s'$.

Hereafter, we propose a structure, called a verifier, which is used to test the codiagnosability of the decentralized system and the fault pattern. The successive steps of the construction are provided in Algorithm 1. Without loss of generality, we assume that $m = 2$, i.e., there are two local sites with their corresponding local projections and local observable event sets.

Algorithm 1: Construction of the verifier.

Input: A pattern Ω and a DFA $G = (L, \Sigma, \delta, l_0)$ with two local sites, where the set of the local observable events is Σ_{o_i}, $i = 1, 2$.

Output: A verifier $G_v = (L_v, \Sigma, \delta_v, l_0^v)$.

1. Construct the synchronous product G_Ω of G and Ω according to Definition 4.
2. Compute a failure diagnoser $G_\mathcal{F}$ that models the faulty behaviors of the system:
   1. Construct the set of states of $G_\mathcal{F}$ as $L_\mathcal{F} = \{(l, s) \in L_{G_\Omega} | \exists w \in \Sigma^* : \delta_{G_\Omega}((l, s), w) \in L_F^{G_\Omega}\}$.
   2. The initial state is $l_0^\mathcal{F} = l_0^{G_\Omega}$ if $l_0^{G_\Omega} \in L_\mathcal{F}$, and undefined otherwise.
   3. For $l_\mathcal{F}, l'_\mathcal{F} \in L_\mathcal{F}$, define the transition function $\delta_\mathcal{F} : L_\mathcal{F} \times \Sigma \to L_\mathcal{F}$ of $G_\mathcal{F}$ as $\delta_\mathcal{F}(l_\mathcal{F}, \sigma) = l'_\mathcal{F}$ if there exists $\sigma \in \Sigma$, such that $\delta_{G_\Omega}(l_\mathcal{F}, \sigma) = l'_\mathcal{F}$.
   4. Define the even set of $G_\mathcal{F}$ as $\Sigma_\mathcal{F} = \{\sigma \in \Sigma | \exists\, l_\mathcal{F}, l'_\mathcal{F} \in L_\mathcal{F} : \delta_\mathcal{F}(l_\mathcal{F}, \sigma) = l'_\mathcal{F}\}$. Then, $G_\mathcal{F} = (L_\mathcal{F}, \Sigma_\mathcal{F}, \delta_\mathcal{F}, l_0^\mathcal{F}, l_F^\mathcal{F})$, where $L_F^\mathcal{F}$ is the set of final state and $L_F^\mathcal{F} = L_F^{G_\Omega}$.
3. Compute a non-failure diagnoser G_η that captures the normal behaviors of the system:
   1. Construct the states set of G_η as $L_\eta = \{(l, s) \in L_{G_\Omega} | (l, s) \in L_{G_\Omega} \setminus L_F^{G_\Omega}\}$.
   2. The initial state is $l_0^\eta = (l_0, s_0)$.
   3. For $l_\eta, l'_\eta \in L_\eta$, define the transition function $\delta_\eta : L_\eta \times \Sigma \to L_\eta$ of G_η as $\delta_\eta(l_\eta, \sigma) = l'_\eta$ if there exists $\sigma \in \Sigma$, such that $\delta_{G_\Omega}(l_\eta, \sigma) = l'_\eta$.
   4. Define the set of events of G_η by $\Sigma_\eta = \{\sigma \in \Sigma | \exists\, l_\eta, l'_\eta \in L_\eta : \delta_\eta(l_\eta, \sigma) = l'_\eta\}$. Then, $G_\eta = (L_\eta, \Sigma_\eta, \delta_\eta, l_0^\eta)$.
4. For $i = 1, 2$, define function $R_i : \Sigma_\eta \to \Sigma_{R_i}$ as

$$R_i(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_{o_i} \text{ or } \sigma \in \Sigma_f, \\ \sigma_{R_i} & \text{if } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f. \end{cases}$$

   Construct local non-failure diagnoser $G_{\eta_i} = (L_\eta, \Sigma_{R_i}, \delta_{\eta_i}, l_0^\eta)$, where $\delta_{\eta_i} : L_\eta \times \Sigma_{R_i} \to L_\eta$ is the transition function defined as $\delta_{\eta_i}(l_\eta, R_i(\sigma)) = l'_\eta$ if there exists $\sigma \in \Sigma$, $l_\eta, l'_\eta \in L_\eta$, such that $\delta_\eta(l_\eta, \sigma) = l'_\eta$.
5. Construct the verifier $G_v = (L_v, \Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma, \delta_v, l_0^v) = G_{\eta_1} || G_{\eta_2} || G_\mathcal{F}$.

Note that for a state $(l_{\eta_1}, l_{\eta_2}, l_\mathcal{F})$ of L_v, it includes three components: l_{η_1}, l_{η_2}, and $l_\mathcal{F}$, where l_{η_1} is a state of G_{η_1}, l_{η_2} is a state of G_{η_2}, and $l_\mathcal{F}$ is a state of $G_\mathcal{F}$. The verifier G_v is constructed by tracking the strings of the local non-failure diagnosers and the failure diagnoser that have the same observation with respect to the local projection $\mathcal{P}_i$, $i = 1, 2$. In other words, the transition relation δ_v of G_v tracks three sequences: one in the local non-failure diagnoser G_{η_1}, one in the local non-failure diagnoser G_{η_2}, and another in the failure diagnoser $G_\mathcal{F}$, which generate the same sequence of observed labels.

In Step 1 of Algorithm 1, we construct a synchronous product of a system G and a pattern Ω, which encodes the system and the pattern. Then, a failure diagnoser $G_\mathcal{F}$ is computed to track the faulty behavior of the system, which is the co-accessible part of the synchronous product with respect to the faulty strings. By Step 3, we build a non-failure diagnoser structure G_η that is the accessible part of the synchronous product by considering the normal behavior of the system. In Step 4, we establish a local non-failure diagnoser G_{η_i} based on the local projection. In this way, the event set $\Sigma_{uo_i} \setminus \Sigma_f$ can be renamed with respect to the local projection and the set of faulty events. Finally, by taking the product of the local non-failure diagnosers and the failure diagnoser, a verifier G_v of the system G and pattern Ω can be set up, which is used to test the codiagnosability of the decentralized system.

Definition 5. *Given the verifier G_v of a DFA G and a pattern Ω, for $0 < m \leq n$, $m, n \in \mathbb{N}$, a cycle $cl : l_v^m, \sigma_m, l_v^{m+1}, \ldots, l_v^n, \sigma_n, l_v^m$ of G_v is said to be an indeterminate cycle if for state $l_v^j = (l_{\eta_1}^j, l_{\eta_2}^j, l_{\mathcal{F}}^j) \in L_v$, $j = m, m+1, \ldots, n$, $l_{\mathcal{F}}^j \in L_F^{\mathcal{F}}$, and there exists σ_j, $j \in \{m, m+1, \ldots, n\}$, such that $\sigma_j \in \Sigma$.*

Theorem 1. *Let G be a DFA with the local projection $\mathcal{P}_i$, $i = 1, 2$, and a pattern Ω. G is not codiagnosable with respect to $\mathcal{P}_i$ and Ω if and only if there exists an indeterminate cycle in G_v.*

Proof. (if) Suppose that there exists an indeterminate cycle $cl : l_v^m, \sigma_m, l_v^{m+1}, \ldots, l_v^n, \sigma_n, l_v^m$. Then, there exists a string $s_v t_v$ generating a run $l_0^v, \sigma_0, \ldots, l_v^m, \sigma_m, l_v^{m+1}, \ldots, l_v^n, \sigma_n, l_v^m$ in G_v, such that for $j = m, m+1, \ldots, n$, $l_{\mathcal{F}}^j \in L_F^{\mathcal{F}}$. Define

$$\mathcal{P}_{R_1} : (\Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma)^* \to \Sigma_{R_1}^*, \tag{3}$$

$$\mathcal{P}_{R_2} : (\Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma)^* \to \Sigma_{R_2}^*, \tag{4}$$

$$\mathcal{P}_{\Sigma} : (\Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma)^* \to \Sigma^*. \tag{5}$$

By $G_v = G_{\eta_1} || G_{\eta_2} || G_{\mathcal{F}}$, it concludes that $\mathcal{L}(G_v) = \mathcal{P}_{R_1}^{-1}(\mathcal{L}(G_{\eta_1})) \cap \mathcal{P}_{R_2}^{-1}(\mathcal{L}(G_{\eta_2})) \cap \mathcal{P}_{\Sigma}^{-1}(\mathcal{L}(G_{\mathcal{F}}))$, where $\mathcal{L}(G_v)$, $\mathcal{L}(G_{\eta_i})$, and $\mathcal{L}(G_{\mathcal{F}})$ are the generated languages of G_v, G_{η_i}, $i = 1, 2$, and $G_{\mathcal{F}}$, respectively. Consequently, there exists a string $st = \mathcal{P}_{\Sigma}(s_v t_v)$ in $G_{\mathcal{F}}$, such that $s = \mathcal{P}_{\Sigma}(s_v) \in \mathcal{L}_{\mathcal{A}}(G)$, $t = \mathcal{P}_{\Sigma}(t_v)$, and $st \in \mathcal{L}_{\mathcal{A}}(G)$ (see Step 2 of Algorithm 1).

Let $s_{R_1} = \mathcal{P}_{R_1}(s_v t_v)$. There exists a string s_1 in G_{η}, such that $\mathcal{P}_{\Sigma}(s_{R_1}) = s_1$ and $\mathcal{P}_{\Sigma}(s_{R_1}) = \mathcal{P}_{\Sigma}(\mathcal{P}_{R_1}(s_v t_v))$. With a slight abuse of notation, we have $\mathcal{P}_1(s_1) = \mathcal{P}_1(st)$. Similar to s_{R_1}, there exists a string s_{R_2} in G_{η_2} and s_2 in G_{η}, such that $s_2 = \mathcal{P}_{\Sigma}(s_{R_2})$. For the same reason, we have $\mathcal{P}_2(s_2) = \mathcal{P}_2(st)$, where $st \in \mathcal{L}_{\mathcal{A}}(G)$, and $s_1, s_2 \notin \mathcal{L}_{\mathcal{A}}(G)$ (see Step 3 of Algorithm 1). For the string $s_v t_v$ with arbitrary length, the strings s_1 and s_2 can also be extended long enough. This implies that G is not codiagnosable with respect to pattern Ω and local projection $\mathcal{P}_i$.

(only if) Suppose that G is not codiagnosable with respect to Ω and $\mathcal{P}_i$. Then, there exists a string $s \in \mathcal{L}_{\mathcal{A}}(G)$, $t \in \mathcal{L}_{\mathcal{A}}(G)/s$, and strings $s_1, s_2 \notin \mathcal{L}_{\mathcal{A}}(G)$, such that $\mathcal{P}_1(s_1) = \mathcal{P}_1(st)$, $\mathcal{P}_2(s_2) = \mathcal{P}_2(st)$.

According to Steps 3 and 4 of Algorithm 1, there exists a string s_{R_1} in G_{η_1} and a string s_{R_2} in G_{η_2}, such that $s_{R_1} = R_1(s_1)$ and $s_{R_2} = R_1(s_2)$, where R can be extended from Σ to Σ^* as the usual way. Consider two prefixes s'_{R_1} of s_{R_1} and s'_{R_2} of s_{R_2}, such that $\mathcal{P}_1(\mathcal{P}_{\Sigma}(s'_{R_1})) = \mathcal{P}_1(s)$ and $\mathcal{P}_2(\mathcal{P}_{\Sigma}(s'_{R_2})) = \mathcal{P}_2(s)$. Since $s_1, s_2 \notin \mathcal{L}_{\mathcal{A}}(G)$, $\mathcal{P}_{\Sigma}(s'_{R_1}), \mathcal{P}_{\Sigma}(s'_{R_2}) \notin \mathcal{L}_{\mathcal{A}}(G)$ holds. As a result, there exist two runs in G_{η_1} and G_{η_2}, beginning from l_0^{η} and generated by s'_{R_1}, s'_{R_2}, respectively, with the forms

$$\rho_{s'_{R_1}} : l_0^{\eta} \xrightarrow{s'_{R_1}} l_{s'_{R_1}}, \text{ and } \rho_{s'_{R_2}} : l_0^{\eta} \xrightarrow{s'_{R_2}} l_{s'_{R_2}}. \tag{6}$$

In addition, since $s \in \mathcal{L}_{\mathcal{A}}(G)$, according to Step 2 of Algorithm 1, there exists a run in $G_{\mathcal{F}}$ beginning from $l_0^{\mathcal{F}}$ generated by s with the form of $\rho_s : l_0^{\mathcal{F}} \xrightarrow{s} l_s$.

Similarly, three runs in G_{η_1}, G_{η_2}, $G_{\mathcal{F}}$ can be generated by s_{R_1}, s_{R_2}, st, respectively, with the form of

$$\rho_{s_{R_1}} : l_0^{\eta} \xrightarrow{s_{R_1}} l_{s_{R_1}}, \ \rho_{s_{R_2}} : l_0^{\eta} \xrightarrow{s_{R_2}} l_{s_{R_2}}, \text{ and } \rho_{st} : l_0^{\mathcal{F}} \xrightarrow{st} l_{st} \tag{7}$$

such that $\mathcal{P}_1(\mathcal{P}_{\Sigma}(s_{R_1})) = \mathcal{P}_1(st)$ and $\mathcal{P}_2(\mathcal{P}_{\Sigma}(s_{R_2})) = \mathcal{P}_1(st)$. Then, there exists a run in G_v of the form

$$\rho_v : l_0^v \to \ldots \to l_v^m \to \ldots \to l_v^n \tag{8}$$

such that $l_v^m = (l_{s'_{R_1}}, l_{s'_{R_2}}, l_s)$ and $l_v^m = (l_{s_{R_1}}, l_{s_{R_2}}, l_{st})$.

The strings s_{R_1} and s_{R_1} are extended from the string st to be as large as possible. Then, there eventually exist two states l_v^i, l_v^j in ρ_v, $m \leq i \leq j \leq n$, such that $l_v^i = l_v^j$. For $r = i, i+1, \ldots, j$, the set of states $\{l_v^r\}$ of G_V form an indeterminate cycle. This contradicts the assumption and ends the proof. □

Theorem 1 provides an approach to verify codiagnosability by searching for the existence of indeterminate cycles. In the case of at least one indeterminate cycle, there are at least three strings s_1, s_2, and s_3 with arbitrary length in non-failure diagnosers G_{η_1}, G_{η_2}, and the failure diagnoser $G_{\mathcal{F}}$, respectively, where s_1 and s_3 have the same observation with respect to the projection $\mathcal{P}_1$, and s_2 and s_3 have the same observation with respect to the projection $\mathcal{P}_2$, violating the codiagnosability.

Example 3. *Consider a DFA G_1 shown in Figure 2 with the local observable event set Σ_{o_i}, $i = 1, 2$, and a pattern Ω shown in Figure 3. According to the first step of Algorithm 1, we construct the synchronous product $G_{1\Omega}$, as shown in Figure 4, which encodes the information of the system G_1 and the fault pattern Ω. The second step is to obtain the failure diagnoser $G_{1\mathcal{F}}$, which is the co-accessible part of $G_{1\Omega}$ with respect to the set of the faulty states $L_F^{G_{1\Omega}}$, as shown in Figure 5a. It should be noted that all accepted behaviors of the failure diagnoser $G_{1\mathcal{F}}$ are faulty behaviors. Continuing Algorithm 1, we compute a non-failure diagnoser $G_{1\eta}$ by taking the accessible part of $G_{1\Omega}$ regarding the set of normal states $L_{G_{1\Omega}} \setminus L_F^{G_{1\Omega}}$, as shown in Figure 5b. Note that all generated behaviors of the non-failure diagnoser $G_{1\eta}$ are normal behaviors. Based on Step 4, we can calculate the local non-failure diagnosers $G_{1\eta_1}$ and $G_{1\eta_2}$, respectively, by renaming the unobservable event sets $\Sigma_{uo_1} \setminus \Sigma_f = \{c\}$ and $\Sigma_{uo_2} \setminus \Sigma_f = \{b\}$ based on function R_i, as shown in Figure 5c,d. It shows that the sets of the events of the local non-failure diagnosers $G_{1\eta_1}$ and $G_{1\eta_2}$ are $\Sigma_{R_1} = \{a, b, f_1, c_{R_1}\}$ and $\Sigma_{R_2} = \{a, b_{R_2}, f_1, c\}$, respectively. The final step of Algorithm 1 is the computation of the verifier G_{1v} of G_1 and Ω, which is obtained by $G_{1v} = G_{1\eta_1} || G_{1\eta_2} || G_{\mathcal{F}}$, as shown in Figure 6.*

According to Theorem 1, one can know that the verification of the fault pattern codiagnosability is to search for the indeterminate cycles in G_{1v}. The verifier of Figure 6 has several cycles (for example, the cycles $7N_27N_22N_2 \xrightarrow{C_{R_1}} 7N_27N_22N_2$, $7N_27N_26F \xrightarrow{C_{R_1}} 7N_27N_26F$, and $7N_27N_26F \xrightarrow{u} 7N_27N_26F$). Notice that only the cycle $7N_27N_26F \xrightarrow{u} 7N_27N_26F$ is an indeterminate cycle (Definition 5). The existence of the indeterminate cycle $7N_27N_26F \xrightarrow{u} 7N_27N_26F$ implies that the system G_1 is not codiagnosable with respect to the local projection $\mathcal{P}_i$, $i = 1, 2$, and the pattern Ω (Theorem 1).

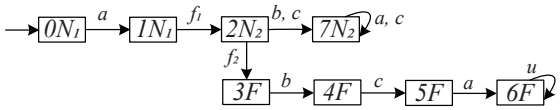

Figure 4. Synchronous product $G_{1\Omega}$ of G_1 and Ω.

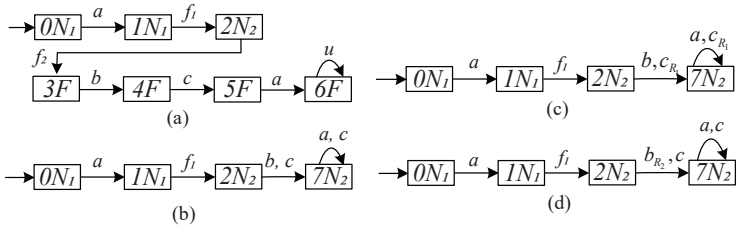

Figure 5. (**a**) Failure diagnoser $G_{1\mathcal{F}}$, (**b**) non-failure diagnoser $G_{1\eta}$, (**c**) local non-failure diagnoser $G_{1\eta_1}$, and (**d**) local non-failure diagnoser $G_{1\eta_2}$.

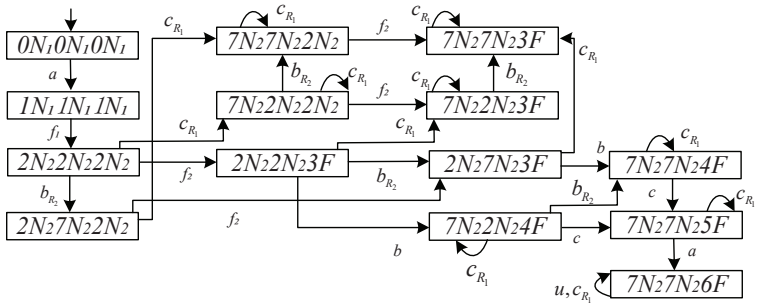

Figure 6. Verifier G_{1v} of G_1 and Ω.

Example 4. *Consider a DFA G_2 shown in Figure 7a with the local projection $\mathcal{P}_i$, $i = 1, 2$ and a pattern Ω in Figure 3, where $\Sigma = \{a, b, c, f_1, f_2\}$ is the set of events, $\Sigma_{o_1} = \{a, b\}$, and $\Sigma_{o_2} = \{a, c\}$. According to the first step of Algorithm 1, we can construct the synchronous product $G_{2\Omega}$, as shown in Figure 7b. Continuing Algorithm 1, the failure diagnoser $G_{2\mathcal{F}}$ can be computed accordingly, as shown in Figure 8a. By Steps 3 and 4 of Algorithm 1, we can obtain the non-failure diagnosers and the local non-failure diagnosers successively. For the sake of simplicity, we keep the local non-failure diagnosers G_{2n_1} and G_{1n_2}, which will be used later, as shown in Figure 8b,c. Continuing Step 5 of Algorithm 1, we can calculate the verifier G_{2v} with respect to the local projections, as shown in Figure 8d.*

Observe that there exist three cycles in the verifier G_{2v}, i.e., $5N_22N_24F \xrightarrow{C_{R_1}} 5N_22N_24F$, $5N_25N_24F \xrightarrow{C_{R_1}} 5N_25N_24F$, and $5N_25N_24F \xrightarrow{C} 5N_25N_24F$, and the cycle $5N_25N_24F \xrightarrow{C} 5N_25N_24F$ is indeterminate (Definition 5). As a consequence, the system G_2 is not codiagnosable with respect to the local projection $\mathcal{P}_i$, $i = 1, 2$, and the pattern Ω (Theorem 1).

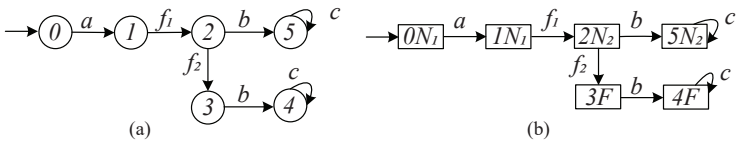

Figure 7. (**a**) Deterministic finite automaton G_2 and (**b**) synchronous product $G_{2\Omega}$.

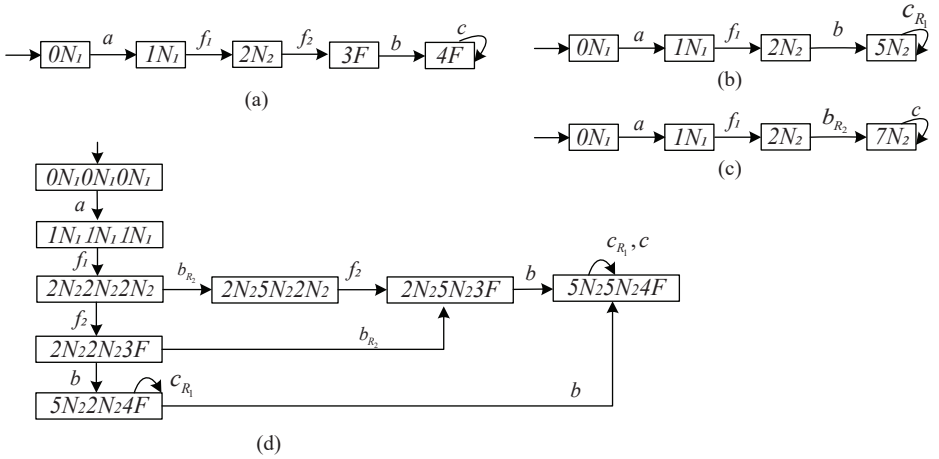

Figure 8. (**a**) Failure diagnoser $G_{2\mathcal{F}}$, (**b**) local non-failure diagnoser G_{2n_1}, (**c**) local non-failure diagnoser G_{2n_2}, and (**d**) verifier G_{2v}.

Example 5. *In order to compare the codiagnosability with the system G_2, we consider a DFA G_3, as shown in Figure 9a, and the pattern Ω in Figure 3. The local projection of G_3 is $\mathcal{P}_i$, $i = 1, 2$, where $\Sigma = \{a, b, c, d, f_1, f_2\}$ is the set of events, $\Sigma_{o_1} = \{a, b, d\}$, and $\Sigma_{o_2} = \{a, c, d\}$. Following Algorithm 1, the resulting structure can be obtained step by step. For simplicity, we detail the*

verifier G_{3v}, as shown in Figure 9b. Note that there is no indeterminate cycle (Definition 5). This implies that the system G_3 is codiagnosable with respect to the local projection and the pattern Ω (Theorem 1).

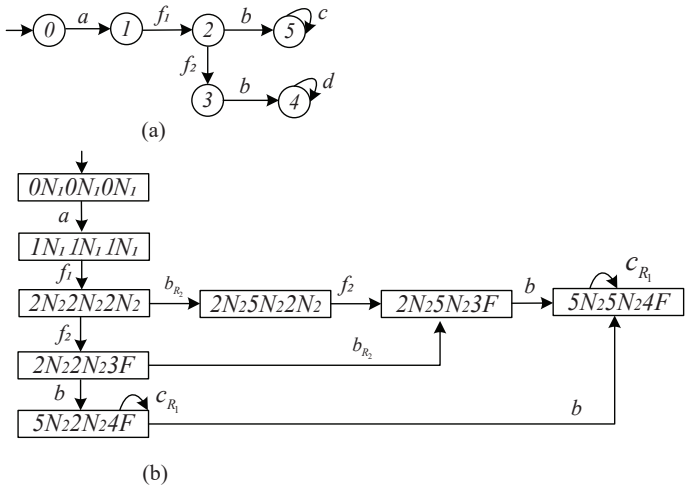

Figure 9. (**a**) Deterministic finite automaton G_3 and (**b**) verifier G_{3v}.

Note that the pattern diagnosability verification in the centralized case can be easily obtained by marking $m = 1$ of Algorithm 1, i.e., the number of the local site is 1. Therefore, the verifier automaton for the centralized case is given as $G_{v_c} = G_{\eta_1} || G_{\mathcal{F}}$, and the necessary and sufficient condition for the non-diagnosability of G is the existence of an indeterminate cycle in G_{v_c}, such that at least one event in the cycle is an event of Σ.

5. Complexity Analysis

From Definitions 1 and 2, we know that the number of states of the system G is $|L|$, and the number of states of the pattern Ω is $|S|$. Assume that the number of the local sites is m.

The complexity of performing Step 1 of Algorithm 1, which constructs the synchronous product G_Ω, is $O(|L| \times |S|)$, and that of Step 2 of Algorithm 1, which constructs the failure diagnoser $G_{\mathcal{F}}$, is $O(|L| \times |S|)$. The complexity of performing Step 3 of Algorithm 1, which computes the non-failure diagnoser G_η by deleting all the final states of G_Ω, is $O(|L| \times |S \setminus \{s_\Omega\}|)$. The complexity of obtaining the local non-failure diagnoser G_{η_i} in Step 4 of Algorithm 1, $i = 1, \ldots, m$, is $O(|L| \times |S \setminus \{s_\Omega\}|)$. By Step 5, the complexity of verifier G_v construction is $O(|L|^m \times |S \setminus \{s_\Omega\}|^m \times |L| \times |S|)$.

Thus, the complexity of Algorithm 1 is $O(|L|^{m+1} \times |S \setminus \{s_\Omega\}|^m \times |S|)$, which is polynomial with respect to the number of the states of G and Ω, i.e., $O((|L||S|)^{m+1})$.

6. Conclusions

The objective of this work is the verification of pattern diagnosability for decentralized DESs. In particular, the fault patterns are modeled by finite automata, providing a general way to formalize different types of failures. This improves the method in [29,31], which only targets single fault scenarios. To this end, we introduce a codiagnosability notion to encapsulate decentralized fault pattern diagnosability, and present an algorithm to test this property. In detail, we first compute a synchronous product structure to encode the system and the pattern. A failure diagnoser and a non-failure diagnoser are calculated based on the synchronous product, where the two structures are obtained by considering the normal and faulty behaviors, respectively. Then, we present a local non-failure diagnoser based on the local projection of the system. A verifier for codiagnosability verification is derived by taking the product of the failure diagnoser with the local non-failure diagnoser. Consequently, the verifier structure can track the strings of the failure and local non-failure diagnosers that have the same observations. The proposed method boasts polynomial

complexity, marking an improvement over methods in [17,21]. Moreover, the approach proposed in this paper can be used for decentralized systems as well as centralized systems, enhancing the method presented in [14,15] for centralized systems.

Our future work will consider decentralized diagnosis issues of timed fault patterns, which are characterized by a sequence of events that occur in a given order at specific values of time or within specific time intervals. In certain cases, the time value of the system is compulsory. For example, in some flexible manufacturing systems, the operation of the robot must be finished in a specified time. At this point, a time value needs to be assigned to each event of the system, and the diagnoser should be calculated based on not only the event but also the time value. Another limitation of the algorithm is that an external attack is not considered in the system. With this in mind, we will consider different types of attack forms in future work, including insertion, deletion, and replacement of observations. These observations refer to sequences of observable events observed by external observers as well as intruders (aliases of attackers). Such tampering with system-generated observations may mislead system operators to make inexact, conservative, or even incorrect state estimations that are critical for many problems in the context of DESs, such as supervisory control, opacity verification, enforcement, detectability analysis, and fault diagnosis. Certainly, these problems can be addressed under the framework of centralized and/or decentralized system architectures.

Author Contributions: Methodology, formal analysis, writing—original draft preparation, Y.L.; supervision, writing—review and editing, G.L.; funding acquisition and programming, A.M.E.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Key R&D Project of China under grant 2018YFB1700104. The authors extend their appreciation to King Saud University, Saudi Arabia, for funding this work through Researchers Supporting Project number (RSP2023R133), King Saud University, Riyadh, Saudi Arabia.

Data Availability Statement: Enquiries about data availability should be directed to the authors.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Cassandras, C.G. The event-driven paradigm for control, communication and optimization. *J. Control Decis.* **2014**, *1*, 3–17. [CrossRef]
2. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*; Springer: New York, NY, USA, 2008.
3. Schuppen, J.; Silva, M.; Seatzu, C. Control of discrete-event systems-Automata and Petri net perspectives. *Lect. Notes Control Inf. Sci.* **2012**, *433*, 319–340.
4. Zaytoon, J.; Lafortune, S. Overview of fault diagnosis methods for discrete event systems. *Annu. Rev. Control* **2013**, *37*, 308–320. [CrossRef]
5. Lafortune, S.; Lin, F.; Hadjicostis, C.N. On the history of diagnosability and opacity in discrete event systems. *Annu. Rev. Control* **2018**, *45*, 257–266. [CrossRef]
6. Lin, F.; Markee, J.; Rado, B. Design and test of mixed signal circuits: A discrete-event approach. In Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993; pp. 217–222.
7. Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; Teneketzis, D. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **1995**, *40*, 1555–1575. [CrossRef]
8. Liang, Y.; Lai, A.; El-Meligy, M.A.; Sharaf, M. Intermittent fault manifestability of discrete event systems. *Soft Comput.* **2023**, *27*, 6999–7009. [CrossRef]
9. Jiang, S.; Huang, Z.; Chandra, V.; Kumar, R. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **2001**, *46*, 1318–1321. [CrossRef]
10. Yoo, T.S.; Lafortune, S. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Trans. Autom. Control* **2002**, *47*, 1491–1495.
11. Zhou, D.; Zhao, Y.; Wang, Z.; He, X.; Gao, M. Review on diagnosis techniques for intermittent faults in dynamic systems. *IEEE Trans. Ind. Electron.* **2019**, *67*, 2337–2347. [CrossRef]
12. Contant, O.; Lafortune, S.; Teneketzis, D. Diagnosis of intermittent faults. *Discret. Event Dyn. Syst.* **2004**, *14*, 171–202. [CrossRef]
13. Biswas, S. Diagnosability of discrete event systems for temporary failures. *Comput. Electr. Eng.* **2012**, *38*, 1534–1549. [CrossRef]
14. Jéron, T.; Marchand, H.; Pinchinat, S.; Cordier, M.O. Supervision patterns in discrete event systems diagnosis. In Proceedings of the 8th International Workshop on Discrete Event Systems, Ann Arbor, MI, USA, 10–12 July 2006; pp. 262–268.

15. Jéron, T.; Marchand, H.; Genc, S.; Lafortune, S. Predictability of sequence patterns in discrete event systems. *IFAC Proc. Vol.* **2008**, *41*, 537–543. [CrossRef]
16. Liang, Y.; Lefebvre, D.; Li, Z. Fault pattern diagnosis of discrete-event systems by means of logical verifiers. *IFAC-PapersOnLine* **2022**, *55*, 551–556. [CrossRef]
17. Genc, S.; Lafortune, S. Diagnosis of patterns in partially-observed discrete-event systems. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; pp. 422–427.
18. Wang, Z.; Wang, J.; Wang, Y. An intelligent diagnosis scheme based on generative adversarial learning deep neural networks and its application to planetary gearbox fault pattern recognition. *Neurocomputing* **2018**, *310*, 213–222. [CrossRef]
19. Watanabe, A.T.; Sebem, R.; Leal, A.B.; Hounsell, M.d.S. Fault prognosis of discrete event systems: An overview. *Annu. Rev. Control* **2021**, *51*, 100–110. [CrossRef]
20. Boussif, A.; Ghazel, M.; Basilio, J.C. Intermittent fault diagnosability of discrete event systems: An overview of automaton-based approaches. *Discret. Event Dyn. Syst.* **2021**, *31*, 59–102. [CrossRef]
21. Gougam, H.E.; Pencolé, Y.; Subias, A. Diagnosability analysis of patterns on bounded labeled prioritized Petri nets. *Discret. Event Dyn. Syst.* **2017**, *27*, 143–180. [CrossRef]
22. Pencolé, Y.; Subias, A. Timed pattern diagnosis in timed workflows: A model checking approach. *IFAC-PapersOnLine* **2018**, *51*, 94–99. [CrossRef]
23. Pencolé, Y.; Subias, A. Diagnosability of event patterns in safe labeled time Petri nets: A model-checking approach. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1151–1162. [CrossRef]
24. Lefebvre, D.; Hadjicostis, C.N. Diagnosability of fault patterns with labeled stochastic Petri nets. *Inf. Sci.* **2022**, *593*, 341–363. [CrossRef]
25. Lefebvre, D.; Li, Z.; Liang, Y. Verifiers for the detection of timed patterns in discrete event systems. *IFAC-PapersOnLine* **2022**, *55*, 264–269. [CrossRef]
26. Lefebvre, D.; Li, Z.; Liang, Y. Diagnosis of timed patterns for discrete event systems by means of state isolation. *Automatica* **2023**, *153*, 111045. [CrossRef]
27. Debouk, R.; Lafortune, S.; Teneketzis, D. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discret. Event Dyn. Syst.* **2000**, *10*, 33–86. [CrossRef]
28. Qiu, W.; Kumar, R. Decentralized failure diagnosis of discrete event systems. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Hum.* **2006**, *36*, 384–395.
29. Wang, Y.; Yoo, T.S.; Lafortune, S. Diagnosis of discrete event systems using decentralized architectures. *Discret. Event Dyn. Syst.* **2007**, *17*, 233–263. [CrossRef]
30. Ye, L.; Daguet, P. A general algorithm for pattern diagnosability of distributed discrete event systems. In Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence, Athens, Greece, 7–9 November 2012; pp. 130–137.
31. Moreira, M.V.; Jesus, T.C.; Basilio, J.C. Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Trans. Autom. Control* **2011**, *56*, 1679–1684. [CrossRef]