Automatica 77 (2017) 93–102

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Computation of minimal diagnosis bases of Discrete-Event Systems using verifiers☆

Leonardo P.M. Santoro, Marcos V. Moreira, João C. Basilio

Universidade Federal do Rio de Janeiro, COPPE-Programa de Engenharia Elétrica, 21949-900, Rio de Janeiro, RJ, Brazil

ARTICLE INFO

Article history:
Received 7 March 2015
Received in revised form
9 September 2016
Accepted 18 October 2016
Available online 13 January 2017

Keywords:
Discrete-event systems
Fault diagnosis
Minimal diagnosis bases
Verifiers

ABSTRACT

In order to diagnose the occurrence of a fault event in a Discrete-Event System (DES), it is first necessary to verify if the language of the system is diagnosable with respect to an observable event set and a fault event set. In some cases, the language of the system is also diagnosable even when a subset of the set of observable events under consideration is used as the actual observable event set. Among the benefits that such a reduction may bring we list the reduction in the number of sensors used in the diagnosis, therefore reducing the cost of the system, and the possibility to deploy the sensor redundancy to obtain a more reliable diagnosis decision. In this work, we propose two algorithms to find, in a systematic way, all minimal subsets of the observable event set that ensure the diagnosability of the DES (minimal diagnosis bases). The methods are based on the construction of verifiers and have lower computational complexity than another method recently proposed in the literature.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

The property of diagnosability of discrete-event systems (DESs) is associated with the ability to detect and isolate the occurrence of an unobservable fault event based on the observation of the traces executed by the system. In Sampath, Sengupta, Lafortune, Sinnamohideen, and Teneketzis (1995), a model-based approach for verifying the diagnosability of DESs is proposed, and a deterministic automaton, called diagnoser, is introduced. Since then, the fault diagnosis problem has been the subject of several works in the literature (Basilio & Lafortune, 2009; Cabral, Moreira, Diene, & Basilio, 2015; Cassez & Tripakis, 2008; Contant, Lafortune, & Teneketzis, 2006; Debouk, Lafortune, & Teneketzis, 2000; Sampath, Lafortune, & Teneketzis, 1998; Zad, Kwong, & Wonham, 2003).

The diagnosis of a fault event depends on the observation of the events executed by the system, and, thus, the choice of sensors to be used is a crucial task in the design of a diagnosis system. In general, it is not necessary to have a complete set of sensors to diagnose the occurrence of a fault event. Moreover, the cost of the diagnosis system increases with the number of sensors used. Therefore, it is important to identify which sensors are redundant for diagnosability purposes, with a view to either removing them or to exploit their redundancy to improve the robustness of the diagnosis system (Carvalho, Moreira, Basilio, & Lafortune, 2013). Although the verification of the diagnosability of the language generated by a DES can be carried out in polynomial time by using verifier automata (Jiang, Huang, Chandra, & Kumar, 2001; Moreira, Basilio, & Cabral, 2016; Moreira, Jesus, & Basilio, 2011; Qiu & Kumar, 2006; Yoo & Lafortune, 2002), the problem of finding the set of observable events with minimum cardinality such that diagnosability holds has been proved to be *NP*-complete (Yoo & Lafortune, 2001).

The simplest way to find the set of observable events with minimum cardinality that ensures language diagnosability, named as minimal diagnosis bases (MDB) in Basilio, Lima, Lafortune, and Moreira (2012), is to perform an exhaustive search in the set 2^{Σ_o}, where Σ_o is the set of potentially observable events of the system. However, the complexity of this task is exponential in the cardinality of Σ_o. In order to mitigate computational efforts, approaches that exploit the *monotonicity* property of diagnosability in static sensor selection problems have been proposed (Debouk, Lafortune, & Teneketzis, 2002; Jiang, Kumar, & Garcia, 2003), and, in

☆ The research work of Marcos Vicente Moreira has been supported by Carlos Chagas Foundation (FAPERJ) (E-26/110.155/2014), by the Brazilian Research Council (CNPq) (309084/2014-8), and by Petrobras (6000.0069294.11.4). The research work of João Carlos Basilio has been supported by the Brazilian Research Council (CNPq), grants 307939/2007-3 and 306592/2010-0. The material in this paper was partially presented at the 12th IFAC-IEEE International Workshop on Discrete Event Systems, May 14–16, 2014, Cachan. This paper was recommended for publication in revised form by Associate Editor Jan Komenda under the direction of Editor Christos G. Cassandras.

E-mail addresses: leosantoro@poli.ufrj.br (L.P.M. Santoro), moreira.mv@poli.ufrj.br (M.V. Moreira), basilio@dee.ufrj.br (J.C. Basilio).

http://dx.doi.org/10.1016/j.automatica.2016.11.026
0005-1098/© 2016 Elsevier Ltd. All rights reserved.

Basilio et al. (2012), an algorithm to compute all MDB is proposed. Although the method presented in Basilio et al. (2012) can be, in several cases, used to mitigate the computational effort of an exhaustive search to find the MDB, the worst-case running time of the algorithm is still high. More recently, Cabasino, Lafortune, and Seatzu (2013) address the optimal sensor selection problem to ensure the diagnosability of DESs modeled by labeled Petri nets whose search is based on the construction of a particular Petri net, called the Verifier Net (Cabasino, Giua, Lafortune, & Seatzu, 2012), although without addressing the problem of finding all MDB. We propose here two methods to find all MDB of DESs modeled by finite state automata. The methods are based on the verifier proposed in Moreira et al. (2011) and, therefore, has smaller computational complexity than the diagnoser approach presented in Basilio et al. (2012).

This paper is organized as follows. In Section 2 we present the necessary background on diagnosability of DESs. In Section 3 we introduce the definition of F and NF-ambiguous cyclic paths, and present an alternative method to obtain a verifier from a previously computed one. In Section 4 (resp. 5) we present the first (resp. second) method for the computation of all MDB of a DES, called the method of the ambiguous cyclic paths (resp. the method of the trees of event sets), and in Section 6, we consider the problem of obtaining all MDB of a DES with multiple fault types. Finally, in Section 7, conclusions are drawn. All proofs of the lemmas and theorems presented here are in Appendix.

2. Theoretical background

2.1. Definitions and notation

Let $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ denote the deterministic automaton model of a DES, where X is the finite state space, Σ is the set of events, $f : X \times \Sigma^\star \to X$ is the transition function, where $\Sigma^\star$ is the Kleene-closure of Σ, $\Gamma : X \to 2^\Sigma$ is the feasible event function, x_0 is the initial state of the system, and X_m is the set of marked states. For the sake of simplicity, the feasible event function and the set of marked states will be omitted unless otherwise stated. The accessible and coaccessible parts of G, denoted as $Ac(G)$ and $CoAc(G)$, respectively, are defined as $Ac(G) = (X_{ac}, \Sigma, f_{ac}, x_0)$, where $X_{ac} = \{x \in X : (\exists s \in \Sigma^\star)[f(x_0, s) = x]\}$ and $f_{ac} : X_{ac} \times \Sigma^\star \to X_{ac}$, and $CoAc(G) = \left(X_{coac}, \Sigma, f_{coac}, x_{0,coac}, X_m\right)$, where $X_{coac} = \{x \in X : (\exists s \in \Sigma^\star)[f(x, s) \in X_m]\}$, $x_{0,coac} = x_0$ if $x_0 \in X_{coac}$, or $x_{0,coac}$ is undefined if $x_0 \notin X_{coac}$, and $f_{coac} : X_{coac} \times \Sigma^\star \to X_{coac}$. The transition functions f_{ac} and f_{coac} are obtained by restricting the domain of f to the accessible and coaccessible states, X_{ac} and X_{coac}, respectively.

The projection $P_s : \Sigma^\star \to \Sigma_s^\star$, where $\Sigma_s \subset \Sigma$, is defined in accordance with Cassandras and Lafortune (2008). It is extended to a language L by applying $P_s(s)$ to all traces $s \in L$. The inverse projection $P_s^{-1} : \Sigma_s^\star \to 2^{\Sigma^\star}$ is defined as $P_s^{-1}(t) = \{s \in \Sigma^\star : P_s(s) = t\}$, and can also be applied to L to obtain $P_s^{-1}(L)$. Let $\bar{K}$ denote the prefix-closure of a language $K \subseteq \Sigma^\star$. Then, K is said to be prefix-closed if $K = \bar{K}$. Notice that, the language generated by an automaton is always prefix-closed. A language $L \subseteq \Sigma^\star$, generated by an automaton G, is said to be live if for all $s \in L$, there exists an event $e \in \Sigma$ such that $se \in L$, *i.e.*, G has no deadlock states.

Let us now suppose that the event set of G is partitioned as $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o and Σ_{uo} denote the sets of observable and unobservable events, respectively, and let $\Sigma_f \subseteq \Sigma_{uo}$ denote the set of fault events. In this paper, we initially assume that there is only one fault event, *i.e.*, $\Sigma_f = \{\sigma_f\}$, and, in Section 6, we remove this assumption. A faulty trace is a sequence of events s such that σ_f is one of the events that form s. A normal trace, on the other hand, does not contain the event σ_f. The set of all normal traces generated by the system is the prefix-closed language $L_N \subset L$. Thus, the set of all traces generated by the system that contain σ_f is $L \setminus L_N$. We will now review the following definitions (Basilio et al., 2012).

Definition 1 (*Path, Cyclic Path, Faulty Path, Elementary Path and Prime Path*)**.**

A. A path in G is a sequence $(x_1, \sigma_1, x_2, \ldots, \sigma_{n-1}, x_n)$, where $x_i \in X, \sigma_i \in \Sigma$, and $x_{i+1} = f(x_i, \sigma_i), i = 1, 2, \ldots, n-1$, and the path is said to be cyclic if $x_1 = x_n$.
B. A faulty path is a path such that at least one of its events is equal to σ_f, *i.e.*, $\sigma_i = \sigma_f$ for some $i \in \{1, \ldots, n-1\}$.
C. A cyclic path $(x_1, \sigma_1, x_2, \ldots, \sigma_{n-1}, x_1)$ is an elementary cyclic path if $x_i \neq x_j, i \neq j, i, j \in \{1, \ldots, n-1\}$.
D. A path $P = (x_1, \sigma_1, x_2, \ldots, x_{l-1}, \sigma_{l-1}, P_l)$ is a prime path if $x_i \neq x_j$, for $i \neq j$, and $i, j \in \{1, \ldots, l\}$, x_1 is the initial state of the system, and P_l is an elementary cyclic path whose initial state is x_l. □

Definition 2 (*Union Product*)**.** The union product of sets Σ_i, $i = 1, 2, \ldots, n$, is defined as follows:

$$\Sigma_1 \dot\times \Sigma_2 \dot\times \ldots \dot\times \Sigma_n = \begin{cases} \{\Sigma_e = \Sigma_{e,1} \cup \Sigma_{e,2} \cup \cdots \cup \Sigma_{e,n} : \\ \quad \Sigma_{e,i} \in \Sigma_i, i = 1, 2, \ldots, n\}, \\ \quad \text{if the elements of } \Sigma_i \text{ are sets,} \\ 2_1^{\Sigma_1} \dot\times 2_1^{\Sigma_2} \dot\times \cdots \dot\times 2_1^{\Sigma_n}, \quad \text{otherwise,} \end{cases}$$

where $2_1^\Sigma = \{\tilde\Sigma \in 2^\Sigma : |\tilde\Sigma| = 1\}$, and $|\cdot|$ denotes cardinality. □

2.2. Diagnosability of discrete event systems

Let G_N denote the subautomaton of G that represents the nonfaulty behavior of the system with respect to the fault event set Σ_f. Then, language diagnosability can be defined as follows.

Definition 3. (Sampath et al., 1995) Let L and $L_N \subset L$ be the live and prefix-closed languages generated by G and G_N, respectively, and define the projection operation $P_o : \Sigma^\star \to \Sigma_o^\star$. Then L is said to be diagnosable with respect to projection P_o and the set of fault events Σ_f if

$$(\exists n \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, \|t\| \geq n) \Rightarrow \\ (\forall \omega \in P_o^{-1}(P_o(st)) \cap L, \omega \in L \setminus L_N),$$

where $\|\cdot\|$ denotes the length of a trace. □

In order to verify the diagnosability of the language generated by the system, several polynomial time methods have been proposed in the literature. We adopt here the verifier structure proposed by Moreira et al. (2011), whose computational complexity is $O(|X|^2 \times |\Sigma|)$. The verifier computed considering Σ_o as the observable event set is denoted in this paper as G_{V,Σ_o}. According to the algorithm proposed in Moreira et al. (2011), the first and second steps are the computation of automata G_N and G_F that model, respectively, the normal and faulty behaviors of G. For the third step, it is necessary to define the renaming function $R : \Sigma_N \to \Sigma_R$, as follows:

$$R(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_o \\ \sigma_R, & \text{if } \sigma \in \Sigma_{uo} \setminus \Sigma_f, \end{cases} \tag{1}$$

which is extended to domain $\Sigma^\star$ as follows: $R(s\sigma) = R(s)R(\sigma)$, for all $s \in \Sigma^\star$ and $\sigma \in \Sigma$, and $R(\epsilon) = \epsilon$. Then, by renaming all unobservable events of G_N, according to Eq. (1), we obtain automaton G_{N_R}. Finally, the verifier is computed as $G_{V,\Sigma_o} = G_{N_R} || G_F = (X_V, \Sigma \cup \Sigma_R, f_V, x_{0,V})$. Notice that each state of G_{V,Σ_o}

is given by $x_V = (x_N, x_F)$, where $x_N = (x, N)$ with $x \in X$, and $x_F = (x, x_{l,NY})$ with $x \in X$ and $x_{l,NY} \in \{N, Y\}$.

The following theorem provides a necessary and sufficient condition for the diagnosability of the language generated by G using the verifier automaton G_{V,Σ_o}.

Theorem 1 (Moreira et al., 2011). *L is not diagnosable with respect to P_o and Σ_f if, and only if, there exists a cyclic path $cl := (x_V^k, \sigma_k, x_V^{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, where $l \geq k > 0$, in G_{V,Σ_o} satisfying the following conditions:*

$$\exists j \in I_{kl} \text{ s.t. for some } x_V^j, (x_{l,NY}^j = Y) \wedge (\sigma_j \in \Sigma), \tag{2}$$

where $I_{kl} = \{k, k+1, \ldots, l\}$.

A cyclic path cl that satisfies condition (2) will be referred to in this paper as an ambiguous cyclic path (Santoro, Moreira, Basilio, & Diene, 2014).

The idea behind the construction of G_{V,Σ_o} is that only the traces of L_N and $\overline{L \setminus L_N}$ that have the same projection are searched. This leads to the following result.

Theorem 2. *A state (x_N, x_F) of G_{V,Σ_o} is reached if, and only if, there exist a trace $s_N \in L_N$ and a trace $s_Y \in \overline{L \setminus L_N}$ such that $P_o(s_Y) = P_o(s_N)$, where x_N and x_F are the states of G_{N_R} and G_F, reached after the occurrence of $s_{N_R} = R(s_N)$ and s_Y, respectively.*

Proof. The proof is straightforward from the proof of Theorem 1 (Moreira et al., 2011). ■

A consequence of Theorem 2, is that the traces s_N and s_Y that lead to the violation of diagnosability of the DES can be easily found from the traces of G_{V,Σ_o}.

2.3. Minimal diagnosis bases

Let us assume that L is diagnosable with respect to projection $P_o : \Sigma^\star \to \Sigma_o^\star$ and Σ_f. Consider the following definitions (Basilio et al., 2012).

Definition 4 (*Diagnosis Basis*). A set $\hat{\Sigma}_o \subseteq \Sigma_o$ is a diagnosis basis for L if L is diagnosable with respect to $\hat{P}_o : \Sigma^\star \to \hat{\Sigma}_o^\star$ and Σ_f. □

Definition 5 (*Minimal Diagnosis Basis*). $\hat{\Sigma}_o \subseteq \Sigma_o$ is a minimal diagnosis basis, if $\hat{\Sigma}_o$ is a diagnosis basis and there is no subset $\tilde{\Sigma}_o \subset \hat{\Sigma}_o$ such that L is diagnosable with respect to $\tilde{P}_o : \Sigma^\star \to \tilde{\Sigma}_o^\star$ and Σ_f. □

From the definitions above we may conclude that every diagnosis basis $\hat{\Sigma}_o \subset \Sigma_o$ is either a minimal diagnosis basis, or it can be obtained from a minimal diagnosis basis by adding to it events from $\Sigma_o \setminus \hat{\Sigma}_o$.

3. Preliminary results

Let $\Sigma_o' \subset \Sigma_o$ be a subset of the set of observable events and consider the problem of verifying if L is diagnosable with respect to $P_o' : \Sigma^\star \to \Sigma_o'^\star$ and Σ_f. If $G_{V,\Sigma_o'}$ has an ambiguous cyclic path, then L is not diagnosable. These ambiguous cyclic paths can be classified as follows.

Definition 6 (*F-Ambiguous Cyclic Path*). An ambiguous cyclic path $(x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^{k+n}, \sigma_{k+n}, x_V^k)$ in verifier $G_{V,\Sigma_o'}$ is an *F*-ambiguous cyclic path if $x_V^{k+j} = (x_N, x_F^{k+j})$ and $\sigma_{k+j} \notin \Gamma_{N_R}(x_N)$, $\forall j \in \{0, \ldots, n\}$. □

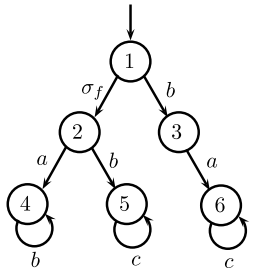

Fig. 1. Automaton model of the system G.

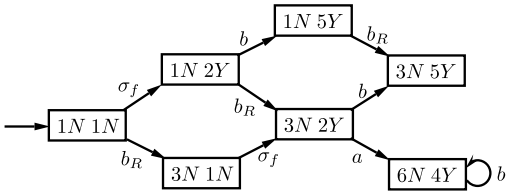

Fig. 2. Verifier $G_{V,\Sigma_o'}$, computed considering $\Sigma_o' = \{a, c\}$.

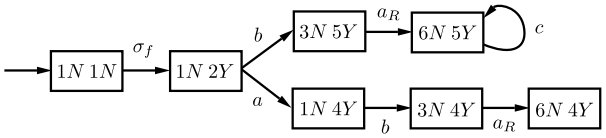

Fig. 3. Verifier $G_{V,\Sigma_o''}$, computed considering $\Sigma_o'' = \{b, c\}$.

Definition 7 (*NF-Ambiguous Cyclic Path*). An ambiguous cyclic path $(x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^{k+n}, \sigma_{k+n}, x_V^k)$ in verifier $G_{V,\Sigma_o'}$ is an *NF*-ambiguous cyclic path if $x_V^{k+j} = (x_N^{k+j}, x_F^{k+j})$, $\forall j \in \{0, \ldots, n\}$, and $\sigma_{k+j} \in \Gamma_{N_R}(x_N^{k+j})$, for some $j \in \{0, \ldots, n\}$. □

Notice that, according to Theorem 1, if L is not diagnosable with respect to P_o' and Σ_f, then there exists an ambiguous cyclic path cl in $G_{V,\Sigma_o'}$, which, in accordance with Theorem 2, is associated with an arbitrarily long length faulty trace $s_Y = st$ and a normal trace s_N, such that $P_o'(s_Y) = P_o'(s_N)$. If the normal trace s_N associated with cl does not have arbitrarily long length, then, according to Definition 6, cl is an *F*-ambiguous cyclic path. On the other hand, if the normal trace s_N associated with cl has arbitrarily long length, then, according to Definition 7, cl is an *NF*-ambiguous cyclic path.

Example 1. Let G be the automaton whose state transition diagram is shown in Fig. 1 with $\Sigma = \{a, b, c, \sigma_f\}$, and let $\Sigma_o = \{a, b, c\}$ be the set of potentially observable events. Verifier $G_{V,\Sigma_o'}$, computed according to Moreira et al. (2011), assuming $\Sigma_o' = \{a, c\}$ as the observable event set, is shown in Fig. 2. According to Definition 6, since $b \notin \Gamma_{N_R}(6N)$, the cyclic path $((6N, 4Y), b, (6N, 4Y))$ is an *F*-ambiguous cyclic path. If we now assume that $\tilde{\Sigma}_o' = \{b, c\}$ is the observable event set, then the resulting verifier is shown in Fig. 3. In this case, since $c \in \Gamma_{N_R}(6N)$, then, according to Definition 7, the cyclic path $((6N, 5Y), c, (6N, 5Y))$ forms an *NF*-ambiguous cyclic path. □

According to Theorem 1, if L is not diagnosable with respect to P_o' and Σ_f, then verifier $G_{V,\Sigma_o'}$ has ambiguous cyclic paths. Thus, with a view to searching for a diagnosis basis for L, events must be added to Σ_o', in order to eliminate the ambiguous cyclic paths of $G_{V,\Sigma_o'}$. In order to avoid the need to construct a new verifier from scratch, we will present an algorithm to compute a verifier $G_{V,\Sigma_o''}$, whose observable event set is $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, where $\sigma \in \Sigma_o \setminus \Sigma_o'$, from verifier $G_{V,\Sigma_o'}$.

Algorithm 1 *Computation of verifier $G_{V,\Sigma_o''}$ from verifier $G_{V,\Sigma_o'}$, where $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$.*

Input: $G_{V,\Sigma_o'} = (X_V', \Sigma_V', f_V', \Gamma_V', x_{0,V}')$

Output: $G_{V,\Sigma_o''} = (X_V'', \Sigma_V' \setminus \{\sigma_R\}, f_V'', \Gamma_V'', x_{0,V}')$

- **Step 1:** *Let Q be a first-in, first-out queue. Add $x_{0,V}'$ to the queue Q and define $X_V'' = \{x_{0,V}'\}$.*
- **Step 2:** *While $Q \neq \emptyset$, do:*
  - **Step 2.1:** *$u \leftarrow Q_1$, where Q_1 is the first entry of queue Q.*
  - **Step 2.2:** *If $\sigma \in \Gamma_V'(u)$ then compute $w = f_V'(u, \sigma)$.*
    - **Step 2.2.1:** *If $\sigma_R \in \Gamma_V'(w)$ then:*
      - ◇ *Create transition $f_V''(u, \sigma) = f_V'(w, \sigma_R)$.*
      - ◇ *If $f_V'(w, \sigma_R)$ has not been added yet to Q, then add it to the end of the queue Q and define $X_V'' = X_V'' \cup \{f_V'(w, \sigma_R)\}$.*
  - **Step 2.3:** *Compute $y = f_V'(u, \hat{\sigma})$, $\forall \hat{\sigma} \in \Gamma_V'(u) \setminus \{\sigma_R, \sigma\}$.*
    - **Step 2.3.1:** *Create transition $f_V''(u, \hat{\sigma}) = y$.*
    - **Step 2.3.2:** *If y has not been added before to Q, then add it to the end of the queue and define $X_V'' = X_V'' \cup \{y\}$.*
  - **Step 2.4:** *Remove the first element of queue Q.*

Example 2. Let G be the automaton whose state transition diagram is shown in Fig. 1 with $\Sigma = \{a, b, c, \sigma_f\}$, and let $\Sigma_o = \{a, b, c\}$ be the set of potentially observable events. Assume we want to compute verifier $G_{V,\Sigma_o''}$ for $\Sigma_o'' = \{a, b, c\}$ from verifier $G_{V,\Sigma_o'}$, shown in Fig. 2, that has been computed assuming $\Sigma_o' = \{a, c\}$. In the first step of Algorithm 1, state $(1N, 1N)$ is inserted in queue Q and also in the set of states of $G_{V,\Sigma_o''}$. The unique event in $\Sigma_V' \setminus \{b_R, b\}$ that is feasible at state $(1N, 1N)$ is σ_f. Then, according to Step 2.3, state $(1N, 2Y)$ is inserted in Q and added to X_V'', and a transition from $(1N, 1N)$ to $(1N, 2Y)$ is created in $G_{V,\Sigma_o''}$ labeled with σ_f. Notice that state $(3N, 1N)$ is not inserted in Q and does not belong to X_V'' since it has been reached from state $(1N, 1N)$ through a transition labeled with b_R. According to Step 2.4, state $(1N, 1N)$ must be removed from Q. Since $Q \neq \emptyset$, Step 2 must be executed again for $u = (1N, 2Y)$. Since event b is feasible in $(1N, 2Y)$, in accordance with Step 2.2.1, it is necessary to check if the state reached from $(1N, 2Y)$ with the transition labeled with b has event b_R in its feasible event set. From Fig. 2, it can be seen that $b_R \in \Gamma_V'(1N, 5Y)$. Thus, a transition from $(1N, 2Y)$ to $(3N, 5Y)$ labeled with b is created, and state $(3N, 5Y)$ is added to the queue and to the state set X_V''. Since state $(3N, 5Y)$ does not have feasible events, then Algorithm 1 is stopped and the verifier $G_{V,\Sigma_o''}$, shown in Fig. 4, is obtained. □

In order to prove the correctness of Algorithm 1, we need the following result.

Lemma 1. *Let $G_{V,\Sigma_o'} = (X_V', \Sigma_V', f_V', \Gamma_V', x_{0,V}')$ and $G_{V,\Sigma_o''} = (X_V'', \Sigma_V' \setminus \{\sigma_R\}, f_V'', \Gamma_V'', x_{0,V}')$ be the verifiers of G with observable event sets Σ_o' and $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, where $\sigma \in \Sigma_o \setminus \Sigma_o'$, respectively. Then $X_V'' \subseteq X_V'$.*

Lemma 1 shows that the states of $G_{V,\Sigma_o''}$ are also states of $G_{V,\Sigma_o'}$. The next lemma shows that if an event e is feasible in a state x_V of $G_{V,\Sigma_o''}$, then it is also feasible in the same state x_V of $G_{V,\Sigma_o'}$.

Lemma 2. *Let $G_{V,\Sigma_o'} = (X_V', \Sigma_V', f_V', \Gamma_V', x_{0,V}')$ and $G_{V,\Sigma_o''} = (X_V'', \Sigma_V' \setminus \{\sigma_R\}, f_V'', \Gamma_V'', x_{0,V}')$ be the verifiers of G with observable event sets Σ_o' and $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, where $\sigma \in \Sigma_o \setminus \Sigma_o'$, respectively, and let $x_V \in X_V''$. Then, $\Gamma_V''(x_V) \subseteq \Gamma_V'(x_V)$.*

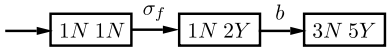

Fig. 4. Verifier $G_{V,\Sigma_o''}$ obtained following the steps of Algorithm 1.

We can now prove the correctness of Algorithm 1.

Theorem 3. *Let $G_{V,\Sigma_o'}$ be the verifier of G with observable event set $\Sigma_o' \subset \Sigma_o$ and let $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$ be the verifiers computed by using the algorithm proposed in Moreira et al. (2011), and Algorithm 1, respectively, considering $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, where $\sigma \in \Sigma_o \setminus \Sigma_o'$, as the observable event set. Then, $G_{V,\Sigma_o''} = \bar{G}_{V,\Sigma_o''}$.*

Remark 1 (*Complexity Analysis of Algorithm 1*)**.** Algorithm 1 is based on the breadth-first search algorithm (Cormen, Leiserson, Rivest, & Stein, 2007) whose complexity is $O(V + E)$, where E and V are the number of edges and vertices of a directed graph, respectively. The complexity of Algorithm 1 is the same as the breadth-first search since each state of the verifier is added to queue Q only once in Step 2, and the verification of Step 2.2.1 has computational complexity $O(|\Sigma|)$ for each state of the verifier. Thus, the complexity of Algorithm 1 is $O(|X|^2 \times |\Sigma|)$. □

4. Method of the ambiguous cyclic paths

As shown in Algorithm 1, the observation of some events eliminates transitions and, therefore, eliminates paths that may have embedded ambiguous cyclic paths. Based on this idea, we present in this section a method to obtain all MDB by eliminating appropriately ambiguous cyclic paths of verifiers computed considering specific sets of observable events. The method will be referred here to as the method of the ambiguous cyclic paths. It has two parts: (i) based on a necessary condition for the diagnosability of L, sets of events that must be subsets of the MDB are found, and (ii) events are appropriately added to those sets in order to find all MDB.

4.1. Elementary diagnosing event sets

Let $x_N \in X_N$ and $x_F \in X_F$ be the components of state $x_V = (x_N, x_F)$ of verifier G_{V,Σ_o}. Form the following sets: $X_{YN} = \{x_V \in X_V : x_F = (x, Y)\}$, and $X_O = \{x_F \in X_F : x_V \in X_{YN}\}$. Notice that X_{YN} is the set of states of verifier G_{V,Σ_o} that are reached by traces that contain σ_f. According to Lemma 1, the states of X_{YN} are also reached with the observation of any set $\Sigma_o' \subset \Sigma_o$, and thus, X_{YN} is a subset of the set of states of verifier $G_{V,\Sigma_o'}$.

Since the language generated by the system is live, each state of X_O is associated with at least one path $P_F = (x_F^k, \sigma_k, x_F^{k+1}, \ldots, x_F^{k+n})$ in G_F, where $x_F^k \in X_O$, satisfying the following conditions: (i) $x_F^{k+n} = x_F^{k+j}$, for some $j \in \{0, 1, \ldots, n-1\}$, *i.e.*, $(x_F^{k+j}, \sigma_{k+j}, \ldots, x_F^{k+n})$ forms a cyclic path; (ii) $(x_F^{k+j}, \sigma_{k+j}, \ldots, x_F^{k+n})$ is the unique cyclic path in P_F. For this reason, set X_O will be referred here to as the post-fault path origin state set and path P_F as the post-fault path. The elements of X_O are called post-fault path origin states.

Definition 8 (*Post-Fault Path Event, Post-Fault Path Event Set*)**.**

A. An event $\sigma \in \Sigma$ is a post-fault path event if it belongs to any post-fault path defined for any state $x_F \in X_O$.
B. A post-fault path event set, denoted as Σ_{fpes}, is the set formed with all events of a post-fault path P_F. □

Let P_F be a post-fault path such that $\Sigma_{fpes} \subseteq (\Sigma \setminus \Sigma_o')$, with $\Sigma_o' \subset \Sigma_o$. The following lemma shows that there exists an F-ambiguous cyclic path in $G_{V,\Sigma_o'}$ associated with post-fault path P_F.

Lemma 3. *Let Σ_{fpes} be the post-fault path event set of $P_F = (x_F^k, \sigma_k, x_F^{k+1}, \ldots, \sigma_{k+n-1}, x_F^{k+n})$, where $x_F^k \in X_O$, and $s_F = \sigma_k\sigma_{k+1}\ldots\sigma_{k+n-1}$ be the sequence of events associated with P_F. Assume that $\Sigma_{fpes} \subseteq (\Sigma \setminus \Sigma_o')$. Then, there exists a faulty path with an embedded F-ambiguous cyclic path in $G_{V,\Sigma_o'}$ whose associated trace is s_F.*

According to Lemma 3, in order to avoid the occurrence of F-ambiguous cyclic paths in $G_{V,\Sigma_o'}$ associated with post-fault paths P_F, at least one event of all post-fault paths must belong to the observable event set Σ_o' of $G_{V,\Sigma_o'}$. The following theorem provides a necessary condition for a set $\Sigma_o' \subset \Sigma_o$ to be a diagnosis basis.

Theorem 4. *Let N_{fpes} denote the number of post-fault paths P_{F_i} of G_F. Then, a necessary condition for $\Sigma_o' \subset \Sigma_o$ to be a diagnosis basis for L and Σ_f is that $\Sigma_o' \cap \Sigma_{fpes,i} \neq \emptyset$, $i = 1, 2, \ldots, N_{fpes}$, where $\Sigma_{fpes,i}$ is the post-fault path event set of P_{F_i}.*

Consider, now, the following definition.

Definition 9 (*Elementary Diagnosing Event Sets*)**.** Let N_{fpes} be number of post-fault path event sets of G_F. The set of all elementary diagnosing event sets is defined as:

$$\Sigma_{edes} = \begin{cases} 2_1^{\Sigma_{fpes,1}}, & \text{if } N_{fpes} = 1, \\ \Sigma_{fpes,1} \dot{\times} \Sigma_{fpes,2} \dot{\times} \ldots \dot{\times} \Sigma_{fpes,N_{fpes}}, & \text{otherwise.} \end{cases}$$

Theorem 4 can be re-stated using Definition 9, as follows.

Theorem 5. *Every diagnosis basis must contain an elementary diagnosing event set.*

The following algorithm provides a systematic way to find all elementary diagnosing event sets from verifier G_{V,Σ_o} and the faulty automaton model G_F.

Algorithm 2 *Computation of the set Σ_{edes} composed of all elementary diagnosing event sets.*

Input: G, Σ_o
Output: Σ_{edes}

- **Step 1:** *Build verifier G_{V,Σ_o} and form sets X_{YN} and X_O. Set $N_{YN} = |X_{YN}|$.*
- **Step 2:** *For each state $x_F^i \in X_O$, associated with a state $(x_N^i, x_F^i) \in X_{YN}$, $i = 1, 2, \ldots, N_{YN}$, verify if $\Gamma_F(x_F^i) \setminus \Gamma_V(x_N^i, x_F^i) \neq \emptyset$. If the answer is no, create an empty tree T_i. If the answer is yes, build a rooted tree T_i from G_F with root x_F^i, as follows:*
  - **Step 2.1:** *Set $\eta_i = |\Gamma_F(x_F^i) \setminus \Gamma_V(x_N^i, x_F^i)|$. Create η_i descendants of x_F^i and label them with $x_{F_n}^i$, $n = 1, \ldots, \eta_i$, where $x_{F_n}^i = f_F(x_F^i, \sigma_n)$, $\sigma_n \in \Gamma_F(x_F^i) \setminus \Gamma_V(x_N^i, x_F^i)$. Label the edge $(x_F^i, x_{F_n}^i)$ with σ_n;*
  - **Step 2.2:** *A node labeled with $x_{F_n}^i$, defined in the tree, will be a leaf if state $x_{F_n}^i$ has already labeled any ancestor of $x_{F_n}^i$. Otherwise, set $\eta_n = |\Gamma_F(x_{F_n}^i)|$ and $x_{F_{new}}^i = f_F(x_{F_n}^i, \sigma_n)$, $\sigma_n \in \Gamma(x_{F_n}^i)$. Create η_n descendants of $x_{F_n}^i$ and label them with $x_{F_{new}}^i$ and the corresponding edge with σ_n. Repeat this step until all states $x_{F_{new}}^i$ give rise only to leaves.*
- **Step 3:** *For each nonempty tree T_i, $i = 1, 2, \ldots, N_{YN}$, identify the leaves $x_{F_l}^i$, $l = 1, \ldots, l_{T_i}$, where l_{T_i} is the number of leaves in tree T_i. Form paths $P_{F_i}^l$, $l = 1, \ldots, l_{T_i}$, starting at x_F^i and ending at $x_{F_l}^i$, $l = 1, \ldots, l_{T_i}$ (these paths are the post-fault paths starting at x_F^i).*
- **Step 4:** *Form the set of events $\Sigma_{edes,i}$ from the post-fault paths $P_{F_i}^l$ obtained in the previous step as follows. If $l_{T_i} = 1$ then set $\Sigma_{edes,i} = 2_1^{\Sigma_{fpes,i}^1}$, else, set $\Sigma_{edes,i} = \Sigma_{fpes,i}^1 \dot{\times} \Sigma_{fpes,i}^2 \dot{\times} \ldots \dot{\times} \Sigma_{fpes,i}^{l_{T_i}}$, where $\Sigma_{fpes,i}^l$ are the post-fault event sets associated with post-fault paths $P_{F_i}^l$, $l = 1, \ldots, l_{T_i}$.*
- **Step 5:** *Form the set Σ_{edes} as follows. If there exists only one nonempty tree T_i, then set $\Sigma_{edes} = 2_1^{\Sigma_{edes,i}}$, else, if there exists more than one nonempty tree, set $\Sigma_{edes} = \Sigma_{edes,i_1} \dot{\times} \Sigma_{edes,i_2} \dot{\times} \ldots \dot{\times} \Sigma_{edes,i_k}$, where $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, N_{YN}\}$. The indices $i_1, \ldots, i_k$ denote the indices of the nonempty trees T_i.*
- **Step 6:** *Remove from Σ_{edes} all sets $\tilde{\Sigma}_{edes} \in \Sigma_{edes}$ for which there exists another set $\hat{\Sigma}_{edes} \in \Sigma_{edes}$ such that $\hat{\Sigma}_{edes} \subset \tilde{\Sigma}_{edes}$.*

In Step 1 of Algorithm 2, sets X_{YN} and X_O are formed from G_{V,Σ_o}. In Steps 2 and 3, the trees with roots in X_O are formed, and all post-fault paths are obtained. In Step 4, the set of events $\Sigma_{edes,i}$, for each nonempty tree T_i, that must belong to $\Sigma_o' \subset \Sigma_o$ to avoid the existence of F-ambiguous cyclic paths in $G_{V,\Sigma_o'}$ associated with a post-fault path $P_{F_i}^l$, $l = 1, \ldots, l_{T_i}$, is computed. In Step 5, the elementary diagnosing event sets are computed by applying the union product to the sets of events $\Sigma_{edes,i}$, associated with the nonempty trees T_i, with a view to guaranteeing that each set of Σ_{edes} has at least one event of each post-fault path event set of G_F. Since these sets are to be used in the search for MDB, those elementary diagnosing event sets that are supersets of another elementary diagnosing event set must be removed from the set, which is performed in Step 6.

Example 3. Let us consider automaton G depicted in Fig. 5 and assume that $\Sigma_o = \{a, b, c, d, e\}$ and $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. According to Algorithm 2, the first step in the computation of the elementary diagnosing event sets is to compute G_{V,Σ_o}, which is shown in Fig. 6, and, based on G_{V,Σ_o}, to form sets $X_{YN} = \{(1N, 2Y), (5N, 3Y), (5N, 4Y)\}$, and $X_O = \{2Y, 3Y, 4Y\}$. In Step 2 of Algorithm 2, for each state of X_O a tree must be built using automaton G_F (not shown in the paper) in order to obtain the post-fault paths with origin at the states of X_O. The trees are shown in Fig. 7. Notice, from Fig. 6, that state $(5N, 3Y)$ is reached from state $(1N, 2Y)$ by the transition labeled with event d, and thus, the tree with root $2Y$ does not have such a transition from $2Y$. For the same reason, the tree with root $3Y$ does not have the transition from $3Y$ labeled with event b.

Continue with Step 3, from the trees of Fig. 7 we find the following post-fault paths: $P_{F_1}^1 = \{2Y, e, 4Y, d, 4Y\}$, $P_{F_1}^2 = \{2Y, e, 4Y, c, 3Y, b, 4Y\}$, $P_{F_1}^3 = \{2Y, e, 4Y, c, 3Y, a, 6Y, b, 3Y\}$, $P_{F_2}^1 = \{3Y, a, 6Y, b, 3Y\}$, $P_{F_3}^1 = \{4Y, d, 4Y\}$, $P_{F_3}^2 = \{4Y, c, 3Y, b, 4Y\}$ and $P_{F_3}^3 = \{4Y, c, 3Y, a, 6Y, b, 3Y\}$. From these paths, we obtain the following post-fault event sets: $\Sigma_{fpes,1}^1 = \{d, e\}$, $\Sigma_{fpes,1}^2 = \{b, c, e\}$, $\Sigma_{fpes,1}^3 = \{a, b, c, e\}$, $\Sigma_{fpes,2}^1 = \{a, b\}$, $\Sigma_{fpes,3}^1 = \{d\}$, $\Sigma_{fpes,3}^2 = \{b, c\}$ e $\Sigma_{fpes,3}^3 = \{a, b, c\}$. According to Steps 4 and 5, the following elementary diagnosing event sets are computed: $\Sigma_{edes} = \{\{a, b, d\}, \{a, b, c, d\}, \{b, d\}, \{b, c, d\}, \{a, b, d, e\}, \{a, b, c, d, e\}, \{b, d, e\}, \{a, c, d, e\}, \{a, c, d\}, \{a, c, d, e\}\}$, which is reduced, according to Step 6, to $\Sigma_{edes} = \{\{b, d\}, \{a, c, d\}\}$. □

4.2. Computation of minimal diagnosis bases

Theorem 5 establishes only a necessary condition for $\Sigma_o' \subset \Sigma_o$ to be a diagnosis basis, and, thus, it is possible that some set $\Sigma_o' \in \Sigma_{edes}$ is not a diagnosis basis for language L. This is so because, the elementary diagnosing event sets computed in Algorithm 2 provide the events that must be observed to avoid the existence of some F-ambiguous cyclic paths in automaton $G_{V,\Sigma_o'}$. However, it is possible that NF-ambiguous cyclic paths and new F-ambiguous cyclic paths may appear. It is, therefore, necessary to identify the remaining ambiguous cyclic paths in $G_{V,\Sigma_o'}$ with a view to eliminating these paths by adding, in an appropriate way, new events to Σ_o'.

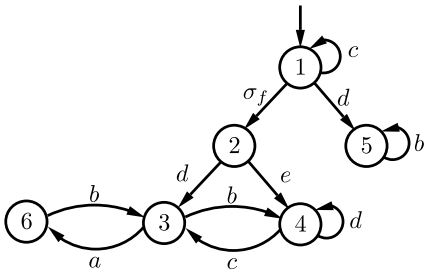

Fig. 5. Automaton G of Example 3.

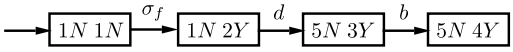

Fig. 6. Verifier G_{V,Σ_o}.

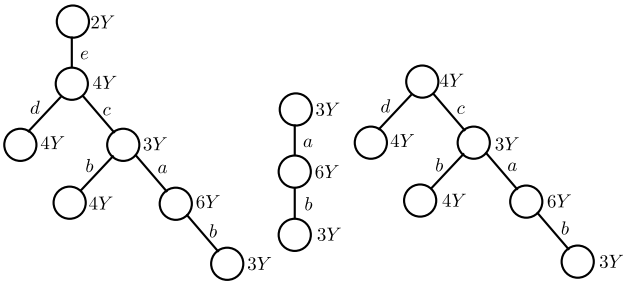

Fig. 7. Trees whose roots are the states of X_0.

As shown in Johnson (1975), all cyclic paths can be decomposed in elementary cyclic paths, and in order to find the elementary ambiguous cyclic paths and identify the events that must be observed to remove such paths, it is first necessary to find the prime paths of $G_{V,\Sigma_o'}$ such that the embedded elementary cyclic path is an ambiguous cyclic path. An algorithm for the computation of all prime paths of an automaton is presented in Basilio et al. (2012). Once the faulty prime paths of $G_{V,\Sigma_o'}$ have been found, it is possible to identify which events must be added to Σ_o' in order to remove the ambiguous cyclic paths. This is the idea behind Algorithm 3 for the computation of all MDB for L.

Algorithm 3 *Computation of the set $\Sigma_{\min}$ of all MDB.*

Input: G, Σ_o
Output: $\Sigma_{\min}$

- **Step 1:** *Apply Algorithm 2 to obtain the elementary diagnosing event sets Σ_{edes}.*
- **Step 2:** *For each elementary diagnosing event set $\Sigma_{edes}^i \in \Sigma_{edes}$, do:*
  - **Step 2.1:** $\Sigma_o' = \Sigma_{edes}^i$.
  - **Step 2.2:** *Build verifier $G_{V,\Sigma_o'}$ considering Σ_o' as the observable event set.*
  - **Step 2.3:** *Verify if L is diagnosable with respect to P_o' and Σ_f by using verifier $G_{V,\Sigma_o'}$. If the answer is yes, Σ_o' is a minimal diagnosis basis. Otherwise, events from $\Sigma_o \setminus \Sigma_o'$ must be added to Σ_o' to form a minimal diagnosis basis.*
  - **Step 2.4:** *Compute all prime paths of $G_{V,\Sigma_o'}$.*
  - **Step 2.5:** *Find the sequences v_j of $G_{V,\Sigma_o'}$, $j = 1, \ldots, n$, associated with the faulty prime paths obtained in Step 2.4 that contain an ambiguous cyclic path, where n denotes the number of faulty prime paths with an embedded ambiguous cyclic path. For each one of the sequences v_j:*
    - **Step 2.5.1:** *Find the normal and faulty traces, $s_N \in L_N$ and $s_F \in L \setminus L_N$ from v_j, respectively.*
    - **Step 2.5.2:** *Find the events $\sigma \in \Sigma_o \setminus \Sigma_o'$ such that $P_o''(s_N) \neq P_o''(s_F)$, where $P_o'' : \Sigma^* \to \Sigma_o''^*$ and $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, and form the set Σ_{new}^j with these events.*
  - **Step 2.6:** *Compute Σ_{new} as follows. If $n = 1$ then $\Sigma_{new} = 2_1^{\Sigma_{new}^1}$, else $\Sigma_{new} = \Sigma_{new}^1 \dot{\times} \Sigma_{new}^2 \dot{\times} \ldots \dot{\times} \Sigma_{new}^n$.*
  - **Step 2.7:** *Remove from Σ_{new} all sets $\tilde{\Sigma}_{new} \in \Sigma_{new}$ for which there exists a set $\hat{\Sigma}_{new} \in \Sigma_{new}$ such that $\tilde{\Sigma}_{new} \supset \hat{\Sigma}_{new}$.*
  - **Step 2.8:** *Let $\Sigma_{\min}^i$ denote a set whose entries are the sets formed by the union of Σ_{edes}^i with each one of the sets of Σ_{new}.*
- **Step 3:** *Compute $\Sigma_{\min} = \Sigma_{\min}^1 \cup \Sigma_{\min}^2 \cup \ldots \cup \Sigma_{\min}^k$, where k is the number of sets of Σ_{edes}.*
- **Step 4:** *Remove from $\Sigma_{\min}$ all sets $\tilde{\Sigma}_{\min} \in \Sigma_{\min}$ for which there exists a set $\hat{\Sigma}_{\min} \in \Sigma_{\min}$ such that $\tilde{\Sigma}_{\min} \supset \hat{\Sigma}_{\min}$.*

The correctness of Algorithm 3 can be proved, as follows.

Theorem 6. *Let G be the automaton model of the system. Then, set $\Sigma_{\min}$, computed by using Algorithm 3, is composed of all minimal diagnosis bases of G.*

Example 4. Let G be the plant automaton shown in Fig. 5, and assume that $\Sigma_o = \{a, b, c, d, e\}$ and $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. The first step of Algorithm 3 is the computation of the elementary diagnosing event sets $\Sigma_{edes} = \{\{a, c, d\}, \{b, d\}\}$. In Step 2, a verifier is computed by considering each set of Σ_{edes} as the observable event set. Let $\Sigma_o' = \{a, c, d\}$. Fig. 8 shows the verifier $G_{V,\Sigma_o'}$, and since $G_{V,\Sigma_o'}$ has an NF-ambiguous cyclic path, then L is not diagnosable with respect to P_o' and Σ_f. From Fig. 9, the following faulty prime path with an embedded NF-ambiguous cyclic path is obtained: $P_{FP} = (\{1N, 1N\}, \sigma_f, \{1N, 2Y\}, e, \{1N, 4Y\}, c, \{1N, 3Y\}, b, \{1N, 4Y\})$. Sequence $v_1 = \sigma_f ecb$ associated with faulty prime path P_{FP} can be obtained, and applying to v_1 the method proposed in Moreira et al. (2011), we obtain the normal and the faulty traces $s_N = c$ and $s_F = \sigma_f ecb$, respectively, such that $P_o'(s_N) = P_o'(s_F)$. Notice that by adding event b or e to the observable event set, *i.e.*, by making $\Sigma_o'' = \Sigma_o' \cup \{b\}$ or $\Sigma_o'' = \Sigma_o' \cup \{e\}$, we have that $P_o''(s_N) \neq P_o''(s_F)$, where $P_o'' : \Sigma^\star \to \Sigma_o''^\star$. Thus, $\Sigma_{new} = \{\{b\}, \{e\}\}$, that results in $\Sigma_{\min}^1 = \{\{a, b, c, d\}, \{a, c, d, e\}\}$, whose elements are MDB.

Proceeding the same way for $\Sigma_o' = \{b, d\}$, we obtain $\Sigma_{\min}^2 = \{\{a, b, c, d\}\}$, and thus, the MDB for L are $\Sigma_{\min} = \Sigma_{\min}^1 \cup \Sigma_{\min}^2 = \{\{a, b, c, d\}, \{a, c, d, e\}\}$. □

4.3. Computational complexity

We will first analyze the computational complexity of Algorithm 2. The first step of Algorithm 2 requires the construction of the verifier automaton G_{V,Σ_o}, according to Moreira et al. (2011), which has complexity $O(|X|^2 \times |\Sigma|)$. After that, for each state $(x_N, x_F) \in X_{YN}$, a tree with root $x_F \in X_O$ must be built. In the worst case, $|X|$ trees are constructed, one for each $x \in X$, associated with a state $x_F = (x, Y)$. Notice that all trees start at a state x_F and so, for each tree, only one node is formed. Since there are $|\Sigma|$ events in G, and G is by assumption deterministic, at most $|\Sigma|$ nodes can be obtained after state x_F. After that, at most $|\Sigma|$ nodes can be obtained for each node in the previous step, resulting in at most $|\Sigma|^2$ nodes. Since there exist $|X|$ states, the maximum number of states in any prime path is $|X|$, which implies that the number of nodes in the tree is upper bounded by $\sum_{i=0}^{|X|} |\Sigma|^i$. Therefore, the complexity of finding all elementary diagnosing event sets by building trees as described in Algorithm 2 is $O(|X| \times |\Sigma|^{|X|})$.

The second step of Algorithm 3 is also based on the construction of trees, then its computational complexity analysis is similar to that carried out in the first step. Notice that a tree is now computed from each verifier automaton obtained for an

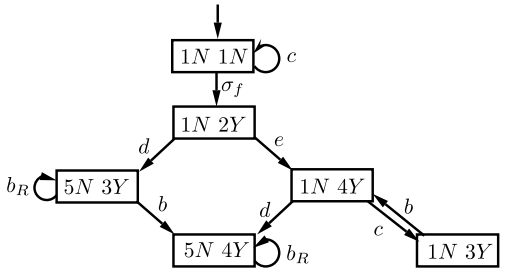

Fig. 8. $G_{V,\Sigma_o'}$ for $\Sigma_o' = \{a, c, d\}$.

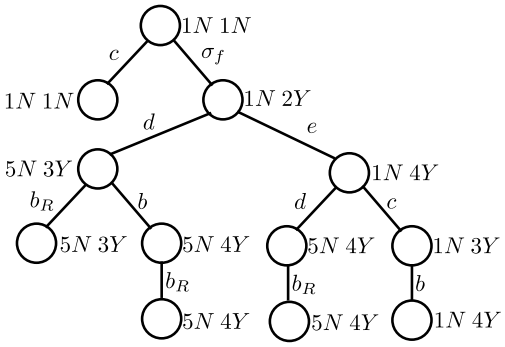

Fig. 9. Tree of $G_{V,\Sigma_o'}$ for $\Sigma_o' = \{a, c, d\}$.

elementary diagnosing event set. Since the verifier can have at most $2|X|^2$ states and at most $2|\Sigma| - 1$ events, then the complexity of constructing a tree in Step 2 is $O(4^{|X|^2} \times |\Sigma|^{2|X|^2})$. Therefore, the overall complexity of Algorithm 3 is $O(|\Sigma_{edes}| \times 4^{|X|^2} \times |\Sigma|^{2|X|^2})$.

The computational complexity of the method presented in Basilio et al. (2012) for the computation of MDB is $O(2^{|\Sigma_o \setminus \Sigma_{edes,\min}|} \times |\Sigma_o|^{|X_d|^2})$, where $\Sigma_{edes,\min}$ is the set with smallest cardinality of Σ_{edes} and X_d is the state-space of the diagnoser which is upper bounded by $2^{|X|}$. This shows that the complexity of Algorithm 3 is smaller than that of the method proposed in Basilio et al. (2012).

It is important to remark that, although the worst-case computational complexity analysis leads to the conclusion that the algorithm proposed in this section is worse than exponential, this is rarely the case, since it will occur only when every state of G and/or the verifier has all events in its feasible event set, and each prime path contains all the states of the graph. For instance, in Example 4, if the exhaustive search method were used to find all MDB, then it would be necessary to compute $2^{|\Sigma_o|} - 2 = 30$ verifiers. On the other hand, following the steps of Algorithm 3, only three verifiers and five trees that do not grow exponentially are needed to compute all MDB. Finally, it is also important to remark that the problem of finding the minimal diagnosis basis with smallest cardinality is NP-complete, and all previous solutions presented in the literature to this problem (Basilio et al., 2012; Cabasino et al., 2013) are also, in the worst-case, worse than exponential.

5. Method of the trees of event sets

We present in this section an algorithm to find all MDB of a DES that avoids the identification of all ambiguous cyclic paths of a verifier. The idea behind the proposed method is as follows. Assume that $|\Sigma_o| = \ell$ and let $\Sigma_o = \{\sigma_{o1}, \sigma_{o2}, \ldots, \sigma_{o\ell}\}$. First, a tree with root $\{\sigma_{o1}\}$ is constructed to find all MDB that contain event σ_{o1}. After that, a second tree is constructed with root $\{\sigma_{o2}\}$ in order to find all MDB that contain event σ_{o2} but do not contain event σ_{o1} since, in the first tree, all MDB that contain event σ_{o1} have already been computed. The procedure continues until the last tree, composed only of root $\{\sigma_{o\ell}\}$, is constructed to verify if $\{\sigma_{o\ell}\}$ is a minimal diagnosis basis.

In the construction of each tree, a verifier is computed associated with each node already found of the tree, and only one ambiguous cyclic path of each verifier must be found in order to obtain the descendants of the nodes in the tree; avoiding, therefore, the computation of all elementary ambiguous cyclic paths of verifiers. For this reason, each tree will be referred to as tree of events.

The proposed method uses recursively the procedure called CREATE-DESCENDANT, presented in Algorithm 5, to create the descendants of a node in a tree of event sets. The main idea of this procedure is to obtain a trace v of the verifier that violates the diagnosability condition, and find all unobservable events in this trace that, if observed, eliminates the ambiguity associated with v. Since the diagnosis bases contain the events that if observed eliminate all ambiguities of the system, then one of the events obtained from v must belong to a diagnosis basis. Variable N and sets Σ_t and Σ_{db}^j used in Algorithms 4 and 5 are global variables.

Algorithm 4 *Computation of* $\Sigma_{\min}$

Input: G, Σ_o
Output: $\Sigma_{\min}$

- **Step 1:** *Set* $N = 0$, $\Sigma_{db} = \emptyset$ *and* $\Sigma_t = \emptyset$.
- **Step 2:** *For each event* $\sigma_o \in \Sigma_o$, *build a tree as follows:*
  - . **Step 2.1:** *Create the root of the tree labeled with* $\{\sigma_o\}$.
  - . **Step 2.2:** *CREATE-DESCENDANT*$(\{\sigma_o\})$ *(Algorithm 5).*
  - . **Step 2.3:** $\Sigma_t \leftarrow \Sigma_t \cup \{\sigma_o\}$.
- **Step 3:** *If* $N = 0$, *then* $\Sigma_{db} = \{\Sigma_o\}$. *Otherwise, form the set* $\Sigma_{db} = \{\Sigma_{db}^1\} \cup \{\Sigma_{db}^2\} \cup \ldots \cup \{\Sigma_{db}^N\}$.
- **Step 4:** *Remove from* Σ_{db} *all sets* $\widetilde{\Sigma}_{db} \in \Sigma_{db}$ *if there exists* $\widehat{\Sigma}_{db} \in \Sigma_{db}$ *such that* $\widehat{\Sigma}_{db} \supseteq \widetilde{\Sigma}_{db}$. *Form the set of MDB* $\Sigma_{\min}$ *with the remaining sets of* Σ_{db}.

Algorithm 5 *CREATE-DESCENDANT*$(\widehat{\Sigma})$

- **Step 1:** $\widetilde{\Sigma}_o \leftarrow \widehat{\Sigma}$.
- **Step 2:** *If* $|\widetilde{\Sigma}_o| = 1$ *compute the verifier* $G_{V,\widetilde{\Sigma}_o}$ *by using the algorithm proposed in Moreira et al. (2011), else use Algorithm 1.*
- **Step 3:** *Verify if there is an ambiguous cyclic path in* $G_{V,\widetilde{\Sigma}_o}$. *If the answer is no:*
  - . **Step 3.1:** $N \leftarrow N + 1$.
  - . **Step 3.2:** *Form the set* $\Sigma_{db}^N = \widetilde{\Sigma}_o$.

  Else:
  - . **Step 3.3:** *Select a sequence* v *associated with a faulty path with an embedded ambiguous cyclic path.*
  - . **Step 3.4:** *Obtain the normal sequence* s_N *and the faulty sequence* st *associated with* v *by using the method proposed in Moreira et al. (2011).*
  - . **Step 3.5:** *For each event* $\sigma_o^i \in \Sigma_o \setminus \widetilde{\Sigma}_o$, *such that* $\widetilde{P}_o(s_N) \neq \widetilde{P}_o(st)$, *where* $\widetilde{P}_o : \Sigma^* \rightarrow (\widetilde{\Sigma}_o \cup \{\sigma_o^i\})^*$ *denotes a projection operation:*
    - **Step 3.5.1:** *Check if the verifier* $G_{V,\widetilde{\Sigma}_o \cup \{\sigma_o^i\}}$ *has been previously constructed.*
    - **Step 3.5.2:** *If* $N > 0$, *check if, for any* $j \in \{1, \ldots, N\}$, $\widetilde{\Sigma}_o \cup \{\sigma_o^i\} \supset \Sigma_{db}^j$.
    - **Step 3.5.3:** *Check if* $\widetilde{\Sigma}_o \cup \{\sigma_o^i\} = \Sigma_o$.
    - **Step 3.5.4:** *Check if* $(\widetilde{\Sigma}_o \cup \{\sigma_o^i\}) \cap \Sigma_t \neq \emptyset$.
  - . **Step 3.6:** *If the answers to Steps 3.5.1 to 3.5.4 are all negative:*

Step 3.6.1: *Create a descendant of the node* $\widetilde{\Sigma}_o$ *in the tree of event sets.*
Step 3.6.2: *Label the node created in Step 3.6.1 with* $(\widetilde{\Sigma}_o \cup \{\sigma_o^i\})$.
Step 3.6.3: *For each node created in Step 3.6.1, execute CREATE-DESCENDANT*$(\widetilde{\Sigma}_o \cup \{\sigma_o^i\})$.

Remark 2. It is important to notice that not all leaves of the trees are diagnosis bases. According to the verifications carried out in Steps 3.5.1 and 3.5.2 of Algorithm 5, a set $\widetilde{\Sigma}_o$ that is not a diagnosis basis will be a leaf of the tree if each set $\widetilde{\Sigma}_o \cup \{\sigma_o^i\}$, obtained in accordance with Step 3.5 of Algorithm 5, contains a diagnosis basis that already labeled one leaf of the tree—this avoids the search from sets that cannot lead to MDB. According to Step 3.5.3 of Algorithm 5, set $\widetilde{\Sigma}_o$ can also be a leaf of a tree, without being a diagnosis basis, when its cardinality is equal to $|\Sigma_o| - 1$ since, by hypothesis, L is diagnosable with respect to Σ_o. Thus, the verification of the diagnosability of L with respect to Σ_o and the creation of a node to represent Σ_o in the tree are unnecessary. It is also important to remark that after the construction of a tree with root $\{\sigma_o\}$, then all the nodes of the other trees cannot have event σ_o, since all diagnosis bases that contain σ_o are necessarily leaves of the tree with root $\{\sigma_o\}$. □

The following theorem proves the correctness of Algorithm 4.

Theorem 7. *The set* $\Sigma_{\min}$*, computed by using Algorithm* 4*, is composed of all minimal diagnosis bases of* G.

Example 5. Let G, shown in Fig. 5, be the automaton model of the system. Suppose that the set of potentially observable events is given by $\Sigma_o = \{a, b, c, d, e\}$ and the set of fault events is $\Sigma_f = \{\sigma_f\}$. We will use now Algorithm 4 to compute the MDB for G.

In accordance with Step 2 of Algorithm 4, we must construct five trees of event sets, each one with the root labeled with an event from Σ_o. Let us consider first the tree with root labeled as $\{a\}$. According to Step 2.2 of Algorithm 4, procedure CREATE-DESCENDANT($\{a\}$) is executed and verifier $G_{V,\{a\}}$, shown in Fig. 10, is computed. Since $G_{V,\{a\}}$ has ambiguous cyclic paths, then $\{a\}$ is not a diagnosis basis. In Step 3.3, a sequence associated with a faulty path with an embedded ambiguous cyclic path must be chosen. Let us choose sequence $v = \sigma_f d_R dbc$. Then, in accordance with Step 3.4, we obtain the normal trace $s_N = d$ and the faulty trace $st = \sigma_f dbc$ associated with v. Since the observation of events b or c makes the projections of s_N and st distinct, then the nodes $\{a, b\}$ and $\{a, c\}$ must be added as descendants of root $\{a\}$. Choosing, now, node $\{a, b\}$, the verifier $G_{V,\{a,b\}}$ is constructed. Since $G_{V,\{a,b\}}$ has ambiguous cyclic paths, then a new sequence associated with a faulty path with an embedded ambiguous cyclic path is chosen and the events that eliminate the ambiguity are used to create new descendants of the node labeled as $\{a, b\}$. This procedure is repeated until either all leaves of the tree are labeled with diagnosis bases or satisfies at least one of the conditions of Step 3.5 of Algorithm 5. The tree with root $\{a\}$, constructed in accordance with Algorithm 4, is shown in Fig. 11. Notice that although the tree has four leaves, namely $\{a, b, c, d\}$, $\{a, b, c, e\}$, $\{a, b, c\}$ and $\{a, c, d, e\}$, only leaves $\{a, b, c, d\}$ and $\{a, c, d, e\}$ are diagnosis bases for L. The sets $\{a, b, c, e\}$ and $\{a, b, c\}$ are not diagnosis bases since, according to Steps 3.5.3 and 3.5.1, respectively, $\{a, b, c, e\}$ has cardinality $|\Sigma_o| - 1$ and the descendant of $\{a, b, c\}$ is a diagnosis bases that has already been discovered.

In accordance with Algorithm 4, the trees with roots labeled with the other four events of Σ_o must also be computed. However, all other trees will be composed only of their roots. Thus, the MDB for L are $\Sigma_{\min} = \{\{a, b, c, d\}, \{a, c, d, e\}\}$.

It is important to remark that, in this example, only 14 verifiers were needed to find all MDB, as opposed to 30 if the exhaustive search method had been used. □

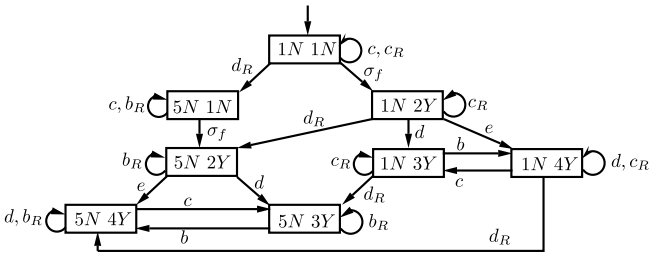

Fig. 10. Verifier $G_{V,\{a\}}$.

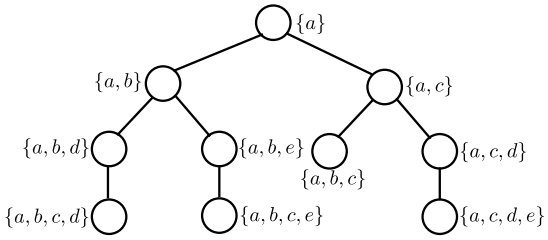

Fig. 11. Tree with root $\{a\}$.

5.1. Complexity analysis

Algorithm 4 requires the computation of $|\Sigma_o|$ trees of event sets, each one with a root labeled with a distinct event of Σ_o. The procedure CREATE-DESCENDANT is executed recursively for each tree until all nodes of the tree are found. In the worst-case, all possible combinations of events of Σ_o will be nodes of the trees. Thus, since the complexity to compute a verifier by using Algorithm 1 is $O(|X|^2|\Sigma|)$, the worst-case complexity of Algorithm 4 is $O(2^{|\Sigma_o|}(|X|^2|\Sigma|))$.

Although the worst-case analysis shows that the complexity of the exhaustive search method is equal to the complexity of Algorithm 4, in general, the computational costs are different, since tests to avoid the creation of unnecessary nodes (Steps 3.5.1–3.5.4), and thus the construction of verifiers, are executed in Algorithm 5.

In comparison with the method of the ambiguous cyclic paths, in the method of the trees of event sets, it is not necessary to compute all elementary ambiguous cyclic paths of a previously computed verifier, being necessary to find only one ambiguous cyclic path for each verifier associated with a node of a tree of event sets, avoiding, therefore, the high computational cost that may exist associated with the search for all elementary ambiguous cyclic paths.

6. Minimal diagnosis bases of DES with multiple fault types

Let $\Sigma_f = \Sigma_{f_1} \dot{\cup} \Sigma_{f_2} \dot{\cup} \ldots \dot{\cup} \Sigma_{f_r}$ be a partition of the set of fault events, where r denotes the number of fault types, and let Π_f denote this partition. The diagnosability of L with respect to Π_f is equivalent to the diagnosability of L with respect to each fault type separately, as ensured by the following result (Yoo & Lafortune, 2002).

Theorem 8. *The language* L *of the system is diagnosable with respect to projection* $P_o : \Sigma^\star \to \Sigma_o^\star$ *and partition* Π_f *on* Σ_f *if, and only if,* L *is diagnosable with respect to* P_o *and* Σ_{f_i}*, for each* $i \in \{1, 2, \ldots, r\}$.

Thus, a diagnosis basis for a DES with multiple fault types is a set of observable events for which the occurrence of a fault event of any type Σ_{f_i} can be diagnosed by its corresponding diagnoser G_{d_i}. Let $\Sigma_{\min,i}$ be the set of all MDB for the ith fault event type, computed by using Algorithm 3 or Algorithm 4. Since each set of $\Sigma_{\min,i}$ contains the events that must be observable to ensure the diagnosability of

the ith fault type, a minimal diagnosis basis for all fault types will be the union of MDB in $\Sigma_{\min,i}$, for $i = 1, \ldots, r$, and can be obtained in accordance with the following algorithm.

Algorithm 6 *Computation of* $\Sigma_{min,t}$

Input: G, Σ_o, $\Sigma_f = \Sigma_{f_1} \dot{\cup} \Sigma_{f_2} \dot{\cup} \ldots \dot{\cup} \Sigma_{f_r}$
Output: $\Sigma_{min,t}$ *: set of all MDB of a DES with multiple fault types*

- **Step 1:** *Compute the set of all MDB* $\Sigma_{min,i}$*, for* $i = 1, \ldots, r$*, by using Algorithm 3 or Algorithm 4.*
- **Step 2:** *Compute* $\Sigma_{min,t} = \Sigma_{min,1} \dot{\times} \Sigma_{min,2} \dot{\times} \ldots \dot{\times} \Sigma_{min,r}$.
- **Step 3:** *Remove from* $\Sigma_{min,t}$ *all sets* $\tilde{\Sigma}_{min,t} \in \Sigma_{min,t}$ *for which there exists another set* $\hat{\Sigma}_{min,t} \in \Sigma_{min,t}$ *such that* $\hat{\Sigma}_{min,t} \subset \tilde{\Sigma}_{min,t}$.

7. Conclusions

We propose in this paper two algorithms to find all minimal diagnosis bases of a discrete-event system, which exploit the structure of verifier automata and are based on the elimination of ambiguous cyclic paths. The methods have smaller computational complexity than another method recently proposed in the literature.

Appendix. Proofs

Proof of Lemma 1. Notice, according to the algorithm proposed in Moreira et al. (2011), that $G_{V,\Sigma_o''} = G''_{N_R} \parallel G_F$, where G''_{N_R} models the non-faulty behavior of the system considering Σ_o'' as the observable event set. Let R'' be the renaming function used to obtain G''_{N_R} from G_N. According to Theorem 2, a state (x_N, x_F) of $G_{V,\Sigma_o''}$ is reached if, and only if, there exist a trace $s_N \in L_N$ and a trace $s_Y \in \overline{L \setminus L_N}$ such that $P_o''(s_Y) = P_o''(s_N)$, and x_N and x_F are, respectively, the states of G''_{N_R} and G_F reached after the occurrence of $s_{N_R} = R''(s_N)$ and s_Y. Since $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, then $P_o'(s_Y) = P_o'(s_N)$, which implies, in accordance with Theorem 2, that the same state (x_N, x_F) is reached in $G_{V,\Sigma_o'}$. Thus, $X_V'' \subseteq X_V'$.

Proof of Lemma 2. Let $G_{V,\Sigma_o'} = G'_{N_R} \parallel G_F$ (resp. $G_{V,\Sigma_o''} = G''_{N_R} \parallel G_F$), where G'_{N_R} (resp. G''_{N_R}) is obtained from G_N by renaming the unobservable events in $\Sigma_N \setminus \Sigma_o'$ (resp. $\Sigma_N \setminus \Sigma_o''$), and let R' (resp. R'') be the renaming function whose unobservable event set is $\Sigma_N \setminus \Sigma_o'$ (resp. $\Sigma_N \setminus \Sigma_o''$). Let $x_V = (x_N, x_F) \in X_V''$. Then, according to Lemma 1, $x_V \in X_V'$. Notice that, the feasible events in $\Gamma_V''(x_V)$ (resp. $\Gamma_V'(x_V)$) can be: (i) renamed events of $\Sigma_R'' = R''(\Sigma_N)$ (resp. $\Sigma_R' = R'(\Sigma_N)$) that are feasible in x_N; (ii) unobservable events in $\Sigma \setminus \Sigma_o''$ (resp. $\Sigma \setminus \Sigma_o'$) that are feasible in x_F; and (iii) events of Σ_o'' (resp. Σ_o') that are feasible in both x_N and x_F. Let $e \in \Gamma_V''(x_V)$. Since $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, then if $e \in \Gamma_V''(x_V) \setminus \{\sigma\}$, it can be seen that $e \in \Gamma_V'(x_V)$. Furthermore, if $e = \sigma$, e is feasible in both x_N and x_F, which implies that $\{\sigma, \sigma_R\} \subseteq \Gamma_V'(x_V)$, therefore concluding the proof.

Proof of Theorem 3. Since $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$ are deterministic automata, it is enough to show that both automata have the same paths.

Let $x_V = (x_N, x_F)$ be a state of $G_{V,\Sigma_o''}$, and suppose that x_V is also a state of $\bar{G}_{V,\Sigma_o''}$. In order to show that $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$ have the same paths, let us consider the problem of obtaining all states of $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$, reached from x_V, after the occurrence of an event. If all reachable states of $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$ are equal, then, since both automata have the same initial state, $G_{V,\Sigma_o''} = \bar{G}_{V,\Sigma_o''}$. According to Algorithm 1, $\bar{G}_{V,\Sigma_o''}$ is computed from $G_{V,\Sigma_o'}$. Thus, in order to prove that the states reached from x_V in $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$ are equal, we must show that the paths of $G_{V,\Sigma_o''}$ can be obtained from the paths of $G_{V,\Sigma_o'}$ in the same way as $\bar{G}_{V,\Sigma_o''}$ is computed. In order to do so, notice, according to Lemma 1, that if x_V belongs to the state space of $G_{V,\Sigma_o''}$, then x_V is also a state of $G_{V,\Sigma_o'}$, which implies, according to Theorem 2, that there exist traces $s_N \in L_N$ and $s_Y \in L \setminus L_N$ such that $P_o''(s_N) = P_o''(s_Y)$, where traces s_N and s_Y lead to states $x_N \in X_N$ and $x_F \in X_F$, respectively. Moreover, since $\Sigma_o'' = \Sigma_o' \cup \{\sigma\}$, then $P_o'(s_N) = P_o'(s_Y)$.

Let $e \in \Gamma_V'(x_V)$, and consider the following cases: (i) $e \in (\Sigma \cup \Sigma_R') \setminus \{\sigma, \sigma_R\}$; (ii) $e \in \{\sigma, \sigma_R\}$. We will show in the sequel that all states of $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$, reached from x_V, are equal for the two cases presented above. Let us first consider case (i), and let (x_V, e, x_{V_f}) be a path of $G_{V,\Sigma_o'}$. Since $e \notin \{\sigma, \sigma_R\}$, then $P_o''(\tilde{e}) = P_o'(\tilde{e})$, where $\tilde{e} = e$, if $e \in \Sigma$, or $\tilde{e} = R'^{-1}(e)$, if $e \in \Sigma_R' \setminus \Sigma$. Thus, according to Theorem 2, path (x_V, e, x_{V_f}) also belongs to $G_{V,\Sigma_o''}$. Notice, according to Step 2.3 of Algorithm 1, that path (x_V, e, x_{V_f}) is also added to $\bar{G}_{V,\Sigma_o''}$. Thus, path (x_V, e, x_{V_f}) belongs to both automata $\bar{G}_{V,\Sigma_o''}$ and $G_{V,\Sigma_o''}$. Let us now consider case (ii), and let $(x_V, e_1, x_{V_1}, e_2, x_{V_f})$ be a path of $G_{V,\Sigma_o'}$. Let us consider four cases: (a) $e_1 = \sigma$ and $e_2 = \sigma_R$; (b) $e_1 = \sigma_R$ and $e_2 = \sigma$; (c) $e_1 = \sigma$ and $e_2 \neq \sigma_R$; (d) $e_1 = \sigma_R$ and $e_2 \neq \sigma$. For cases (a) and (b), it can be seen that $P_R'(ve_1e_2) = s_{N_R}\sigma_R$ and $P'(ve_1e_2) = s_Y\sigma$, where $P_R' : (\Sigma \cup \Sigma_R')^\star \to \Sigma_R'^\star$ and $P' : (\Sigma \cup \Sigma_R')^\star \to \Sigma^\star$. Thus, since $\sigma \in \Sigma_o''$, $P_o''(R'^{-1}(s_{N_R}\sigma_R)) = P_o''(s_Y\sigma)$, which implies, according to Theorem 2, that path (x_V, σ, x_{V_f}) belongs to verifier $G_{V,\Sigma_o''}$, and, according to Step 2.2 of Algorithm 1, the same path also belongs to verifier $\bar{G}_{V,\Sigma_o''}$. Finally, let us consider cases (c) and (d) and define $\tilde{s}_{N_R} = P_R'(ve_1e_2)$ and $\tilde{s}_Y = P'(ve_1e_2)$. In these cases, it can be seen that $P_o''(R'^{-1}(\tilde{s}_{N_R})) \neq P_o''(\tilde{s}_Y)$, which implies that event σ is not feasible in state x_V of $G_{V,\Sigma_o''}$. The same occurs in automaton $\bar{G}_{V,\Sigma_o''}$ as it can be seen in Step 2.2 of Algorithm 1. Since, according to Lemma 2 and Algorithm 1, no other transition from state x_V in $G_{V,\Sigma_o''}$ and $\bar{G}_{V,\Sigma_o''}$, respectively, may exist, we can conclude that $G_{V,\Sigma_o''} = \bar{G}_{V,\Sigma_o''}$.

Proof of Lemma 3. Let Σ_R' be the event set of automaton G'_{N_R} obtained considering Σ_o' as the observable event set. Since, by assumption, $\Sigma_{fpes} \subseteq (\Sigma \setminus \Sigma_o')$, then $\Sigma_{fpes} \cap \Sigma_R' = \emptyset$. Therefore, the events of Σ_{fpes} are private events of automaton G_F. Since $G_{V,\Sigma_o'} = G'_{N_R} \parallel G_F$ and $x_F^k \in X_O$, there exists a faulty path with an embedded cyclic path in verifier $G_{V,\Sigma_o'}$ whose associated trace is s_F, which, according to Definition 6, is an F-ambiguous cyclic path.

Proof of Theorem 4. Suppose that there exists $i \in \{1, 2, \ldots, N_{fpes}\}$ such that $\Sigma_o' \cap \Sigma_{fpes,i} = \emptyset$. Then, there exists a post-fault path $P_{F_i} = (x_F^k, \sigma_k, x_F^{k+1}, \ldots, x_F^{k+m})$ such that $\Sigma_{fpes,i} \subseteq (\Sigma \setminus \Sigma_o')$. Thus, in accordance with Lemma 3, there exists an F-ambiguous cyclic path in the verifier automaton $G_{V,\Sigma_o'}$.

Proof of Theorem 6. In Step 1 all elementary diagnosing event sets $\Sigma_{edes}^i \in \Sigma_{edes}$, $i = 1, \ldots, k$, are computed, since, according to Theorem 5, every diagnosis basis must contain an elementary diagnosing event set. In Steps 2.4 and 2.5, all prime paths with an embedded ambiguous cyclic path, v_j, of G_{V,Σ_{edes}^i}, for $j = 1, \ldots, n$, are obtained, and the events that solve the ambiguity between the normal and the faulty trace associated with v_j are stored in the event set Σ_{new}^j. In Step 2.6, the set Σ_{new}, whose elements are sets of events whose observations eliminate all ambiguous cyclic paths of G_{V,Σ_{edes}^i}, is computed, and in Step 2.7, the elements of Σ_{new} that contain redundant events are removed from Σ_{new}, since only MDB are being searched. In Step 2.8, diagnosis bases that contain Σ_{edes}^i are obtained, forming the set $\Sigma_{\min}^i$. Notice that all diagnosis bases that contain Σ_{edes}^i are obtained from the diagnosis bases of $\Sigma_{\min}^i$ by adding new events to them. Moreover, the elements of $\Sigma_{\min}^i$ are the diagnosis bases with smallest cardinality that contain Σ_{edes}^i. In

Step 3, the set $\Sigma_{\min}$ is formed by the union of all sets of diagnosis bases obtained in Step 2. Thus, all diagnosis bases can be obtained from the diagnosis bases of $\Sigma_{\min}$. Finally, in Step 4, to ensure that all elements of $\Sigma_{\min}$ are MDB, the sets that contain redundant events are eliminated from $\Sigma_{\min}$.

Proof of Theorem 7. We will prove Theorem 7 in three steps. We first prove that the sets of Σ_{db} are diagnosis bases for G, then we prove that a minimal diagnosis basis labels at least one leaf of a tree of event sets constructed in Algorithm 4, and finally, we show that all sets of $\Sigma_{\min}$, computed by using Algorithm 4, are MDB.

Part 1: *The sets in Σ_{db} are diagnosis bases.* Notice that Algorithm 4 is based on the construction of verifier automata and that, in accordance with Step 3.2 of Algorithm 5, an event set Σ_{db}^{j} is formed only when the verifier $G_{V,\Sigma_{db}^{j}}$ does not have ambiguous cyclic paths. Thus, all sets that belong to Σ_{db} are diagnosis bases.

Part 2: *Σ_{db} contains all MDB.* Then, according to Algorithm 4, $|\Sigma_o|$ trees of event sets will be constructed, each one with a root labeled with a distinct event of Σ_o. Let $\Sigma_{db}^{j} = \{\sigma_1, \sigma_2, \ldots, \sigma_k\} \subset \Sigma_o$ be a minimal diagnosis basis for L, and consider, without loss of generality, the tree with root labeled with σ_1. Since Σ_{db}^{j} is a minimal diagnosis basis, then there exists at least one sequence v in the language generated by $G_{V,\{\sigma_1\}}$ associated with an ambiguous cyclic path. According to Step 3.3 of Algorithm 5, a sequence v associated with an ambiguous cyclic path in $G_{V,\{\sigma_1\}}$ is chosen to obtain the descendants of node $\{\sigma_1\}$ in its respective tree. Since Σ_{db}^{j} is a minimal diagnosis basis, the simultaneous observation of the events $\sigma_1, \sigma_2, \ldots, \sigma_k$ delete all the ambiguous cyclic paths in $G_{V,\{\sigma_1\}}$, including the one that is associated with v and, therefore, at least one of the events in $\Sigma_{db}^{j} \setminus \{\sigma_1\}$, together with σ_1, is a descendant of $\{\sigma_1\}$ in the tree with root $\{\sigma_1\}$. The algorithm continues until all ambiguous cyclic paths have been removed. From the definition of minimal diagnosis basis, any subset of Σ_{db}^{j} is not a diagnosis basis, and, thus, at least one leaf of the tree of events with root $\{\sigma_1\}$ must be labeled with Σ_{db}^{j}, and according to Step 3 of Algorithm 4, Σ_{db}^{j} will be added to Σ_{db}.

Part 3: *The sets of $\Sigma_{\min}$ are MDB.* Notice that, in Step 4 of Algorithm 4, all leaves created with events that are not essential to the diagnosis of the fault event are deleted, ensuring that only MDB remain in $\Sigma_{\min}$.

References

Basilio, J.C., & Lafortune, S. (2009). Robust codiagnosability of discrete event systems. In *Proc 2009 American control conference*. St. Louis, MO, (pp. 2202–2209).

Basilio, J. C., Lima, S. T. S., Lafortune, S., & Moreira, M. V. (2012). Computation of minimal event bases that ensure diagnosability. *Discrete Event Dynamic Systems: Theory and Applications*, *22*(3), 249–292.

Cabasino, M. P., Giua, A., Lafortune, S., & Seatzu, C. (2012). A new approach for diagnosability analysis of Petri nets using verifiers. *IEEE Transactions on Automatic Control*, *57*(12), 3104–3117.

Cabasino, M. P., Lafortune, S., & Seatzu, C. (2013). Optimal sensor selection for ensuring diagnosability in labeled Petri nets. *Automatica*, *49*(8), 2373–2383.

Cabral, F. G., Moreira, M. V., Diene, O., & Basilio, J. C. (2015). A Petri net diagnoser for discrete-event systems modeled by finite state automata. *IEEE Transactions on Automatic Control*, *60*(1), 59–71.

Carvalho, L. K., Moreira, M. V., Basilio, J. C., & Lafortune, S. (2013). Robust diagnosis of discrete event systems against permanent loss of observations. *Automatica*, *49*(1), 223–231.

Cassandras, C., & Lafortune, S. (2008). *Introduction to discrete event system*. Secaucus, NJ: Springer-Verlag New York, Inc..

Cassez, F., & Tripakis, S. (2008). Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, *88*, 497–540.

Contant, O., Lafortune, S., & Teneketzis, D. (2006). Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems: Theory and Applications*, *16*(1), 9–37.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2007). *Introduction to algorithms*. Massachusetts: MIT Press.

Debouk, R., Lafortune, S., & Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, *10*(1), 33–86.

Debouk, R., Lafortune, S., & Teneketzis, D. (2002). On an optimization problem in sensor selection. *Discrete Event Dynamic Systems: Theory and Applications*, *12*(4), 417–445.

Jiang, S., Huang, Z., Chandra, V., & Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, *46*(8), 1318–1321.

Jiang, S., Kumar, R., & Garcia, H. (2003). Optimal sensor selection for discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, *48*(3), 369–381.

Johnson, D. B. (1975). Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 77–84.

Moreira, M. V., Basilio, J. C., & Cabral, F. G. (2016). "Polynomial time verification of decentralized diagnosability of discrete event systems" vs. "Decentralized failure diagnosis of discrete event systems": A critical appraisal. *IEEE Transactions on Automatic Control*, *61*(1), 178–181.

Moreira, M. V., Jesus, T. C., & Basilio, J. C. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, *56*(7), 1679–1684.

Qiu, W., & Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, *36*(2), 384–395.

Sampath, M., Lafortune, S., & Teneketzis, D. (1998). Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, *43*(7), 908–929.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, *40*(9), 1555–1575.

Santoro, L.P.M., Moreira, M.V., Basilio, J.C., & Diene, O. (2014). Computation of minimal diagnosis bases of discrete-event systems using verifiers: Method of the ambiguous cyclic paths. In *12th IFAC - IEEE international workshop on discrete event systems*. Paris, France, (pp. 440–445).

Yoo, T.-S., & Lafortune, S. (2001). On the computational complexity of some problems arising in partially-observed discrete-event systems. In *Proc 2001 American control conference*. Arlington, VA, (pp. 307–312).

Yoo, T.-S., & Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, *47*(9), 1491–1495.

Zad, S. H., Kwong, R. H., & Wonham, W. M. (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control*, *48*(7), 1199–1212.

Leonardo P. M. Santoro was born in Rio de Janeiro, Brazil, in 1986. He received the Electrical Engineering degree and the M.Sc. degree in Automation and Control from the Federal University of Rio de Janeiro, in 2009 and 2013, respectively. Since 2009, he has been working as a Control Instrumentation Team Leader at the Brazilian company Radix Engenharia e Software.

Marcos V. Moreira was born on May 11, 1976 in Rio de Janeiro, Brazil. He received the Electrical Engineer degree, the M.Sc. degree and the D.Sc. degree in Control from the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 2000, 2002 and 2006, respectively. Since 2007, he has been an Associate Professor at the Department of Electrical Engineering at the Federal University of Rio de Janeiro. His main interests are robust failure diagnosis of discrete-event systems, cyber-attacks, smart grids, and the development of control laboratory techniques.

João C. Basilio was born on March 15, 1962 in Juiz de Fora, Brazil. He received the Electrical Engineering degree in 1986 from the Federal University of Juiz de Fora, Juiz de Fora, Brazil, the M.Sc. degree in Control from the Military Institute of Engineering, Rio de Janeiro, Brazil, in 1989, and the Ph.D. degree in Control from Oxford University, Oxford, UK, in 1995. He began his career in 1990 as an Assistant Lecturer at the Department of Electrical Engineering of the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, where he is currently a Full Professor in Control. Since February 2014, he has been the Dean of Polytechnic School of UFRJ. From September 2007, to December 2008, he spent a sabbatical leave at the University of Michigan, Ann Arbor, and was an Invited Professor of École Centrale of Lille, University of Lille, France, during September 2016. His current interests are fault diagnosis and supervisory control of discrete-event systems. Prof. Basilio is the recipient of the Correia Lima Medal.