Optimal selection of subsystems for synchronous diagnosis [⋆]

Lucas N. R. Reis [*] **Marcos V. Moreira** [*]

[*] *COPPE - Electrical Engineering Program, Universidade Federal do Rio de Janeiro, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil, (lucasnrreis@poli.ufrj.br/moreira.mv@poli.ufrj.br).*

Abstract: Fault diagnosis of automated systems is a very important task, since faults can alter the expected behavior of systems, damaging equipment and bringing risk to operators. In general, systems are composed of several subsystems or modules, and therefore, the complete system model may grow exponentially with the number of system components. This fact shows that a large amount of memory space may be needed to implement diagnosers computed using traditional methods, since they are based on the complete system model. Recently, a new method for fault diagnosis, called synchronous diagnosis, has been proposed. The diagnoser computed using this method is based on the state observers of the fault-free component models of the system, avoiding the implementation of the state observer of the composed system model. In the synchronous diagnosis strategy it is supposed that all fault-free subsystem models are used to detect the fault occurrence. However, in practice, some subsystems may not add useful information regarding the fault occurrence, or the same information can be obtained from the other modules, which shows that these subsystems are not necessary in the synchronous diagnosis scheme. In this paper, an algorithm for computing all minimal sets of modules that ensure the synchronous diagnosability of a Discrete Event System is proposed. The performance of the proposed algorithm is compared with the performance of the exhaustive search method, and we show that using the proposed method there is a significant reduction in the computational cost of finding all minimal sets of modules that ensure synchronous diagnosability.

Keywords: Fault Diagnosis, Diagnosability, Discrete Event Systems, Synchronous Diagnosis, Verifiers.

1. INTRODUCTION

Fault diagnosis of automated systems is a very important task, since faults can alter the expected behavior of systems, damaging equipment and bringing risk to operators. This problem is addressed in several works in the literature (Sampath et al., 1995, 1996; Zad et al., 2003; Basile et al., 2009; Fanti et al., 2013; Gascard and Simeu-Abazi, 2013; Cabasino et al., 2014; Basile et al., 2017). In the seminal work Sampath et al. (1995), a diagnoser automaton is proposed to perform fault diagnosis and to verify the diagnosability of the system language, *i.e.*, to verify if the fault occurrence can be detected within a bounded number of event occurrences after the fault. The main problem with respect to the solution presented in Sampath et al. (1995), is that the diagnoser is constructed based on an observer automaton, whose computational complexity may grow exponentially with the number of system states.

In order to overcome the exponential problem for verifying the diagnosability of the system language, in Moreira et al. (2011) it is proposed a different strategy based on a verifier automaton that can be computed in polynomial time. However, the verifier cannot be straightforwardly used for online diagnosis. In Cabral et al. (2015), a fault diagnosis strategy that avoids the implementation of the diagnoser presented in Sampath et al. (1995) is proposed. Instead of implementing the complete observer automaton of the system, the state estimate is computed online, and the size of the diagnoser is polynomial with respect to the size of the plant automaton G.

Although in Cabral et al. (2015) the size of the diagnoser is reduced in comparison with the classical diagnoser proposed in Sampath et al. (1995), it still has an exponential growth with respect to the number of system components. This is due to the fact that the diagnoser is obtained from the complete plant model, which is, in general, computed from the parallel composition of subsystems or components. In order to circumvent this problem, in Debouk et al. (2002) and Contant et al. (2006), notions of modular diagnosability are proposed, where the idea is to infer the occurrence of the fault event by observing only the local component where the fault is modeled. It is important to remark that in the modular diagnosis techniques, the following two assumptions are considered: (i) all common events between subsystems are observable; and (ii) the component where the fault event is modeled has persistent excitation, *i.e.*, the fault does not bring the component,

[⋆] This work has been partially supported by FAPERJ under grant E-26/211.136/2019, the Brazilian Research Council (CNPq) under grants 305267/2018-3 and 431307/2018-0, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001.

and thus the system, to a halt. Note that, according to these assumptions, the system modules cannot be synchronized with unobservable events, which implies that the fault event cannot be modeled in more than one system module. In addition, it is necessary to guarantee the persistence of excitation property, which requires the previous knowledge of the system behavior.

In order to relax all assumptions considered in the modular diagnosis strategy, a new diagnosis technique, called synchronous diagnosis, is proposed in Cabral and Moreira (2020). The method relies on the computation of a diagnoser based on the state observers of the fault-free component models of the system, avoiding the implementation of the state observer of the composed system model. In the synchronous diagnosis strategy it is supposed that all fault-free subsystem models are used to detect the fault occurrence. However, in practice, some subsystems may not add useful information regarding the fault occurrence, or the same information can be obtained from the other modules, which implies that these modules are not necessary in the synchronous diagnosis scheme. It is important to remark that finding the minimum number of system modules necessary to diagnose the fault occurrence, reduces the size of the diagnoser and the memory space necessary to store it in a computer. The computation of the useful components or subsystems for fault diagnosis is not carried out in Cabral and Moreira (2020).

The simplest way to find all minimal subsets of modules that ensure language synchronous diagnosability is to perform an exhaustive search, computing the verifier for all $2^r - 1$ possible subsets of modules, where r denotes the number of system modules, and selecting those that have smaller cardinality and do not contain another subset of modules. This procedure has a high computational cost. Thus, in this paper, we present a method to compute all minimal sets of modules that are necessary to guarantee the synchronous diagnosability of the system language, that, in general, does not require the computation of the verifier for all subsets of system modules. After that, the minimum cardinality sets can be obtained simply by choosing those that have smaller cardinality.

This paper is organized as follows. In Section 2, we present some preliminary concepts and the definitions of diagnosability and synchronous diagnosability. In Section 3, we provide an algorithm for the computation of all minimal subsets of modules for synchronous diagnosability. In Section 4, we compare the computational cost of the proposed algorithm with the exhaustive search method for an example, where we show that the proposed method can lead to a significant reduction in computation and time. The conclusions are drawn in Section 5.

2. PRELIMINARIES

2.1 Notations and Definitions

Let $G = (X, \Sigma, f, x_0, X_m)$ be the automaton model of a DES, where X is the set of states, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the transition function, which can be partially defined over its domain, x_0 is the initial state, and X_m is the set of marked states. The set of marked states will be omitted unless otherwise stated. The domain of the transition function f can be extended to $X \times \Sigma^*$, where Σ^* denotes the Kleene-closure of Σ, as usual: $f(x, \varepsilon) = x$, and $f(x, s\sigma) = f(f(x, s), \sigma)$, for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, where ε denotes the empty sequence. The language generated by G is defined as $\mathcal{L}(G) = \{s \in \Sigma^* : f(x_0, s)!\}$, where $f(x_0, s)!$ denotes that $f(x_0, s)$ is defined.

The accessible and coaccessible parts of G, denoted by $Ac(G)$ and $CoAc(G)$, respectively, are defined as: $Ac(G) = (X_{ac}, \Sigma, f_{ac}, x_0, X_{m,ac})$, where $X_{ac} = \{x \in X : (\exists s \in \Sigma^*)[f(x_0, s) = x]\}$, $f_{ac} : X_{ac} \times \Sigma \to X_{ac}$, and $X_{m,ac} = X_m \cap X_{ac}$, and $CoAc(G) = (X_{coac}, \Sigma, f_{coac}, x_{0coac}, X_m)$, where $X_{coac} = \{x \in X : (\exists s \in \Sigma^*)[f(x, s) = X_m]\}, x_{0coac} = x_0$ if $x_0 \in X_{coac}$, or x_{0coac} is undefined if $x_o \notin X_{coac}$, and $f_{coac} : X_{coac} \times \Sigma \to X_{coac}$. The transition functions f_{ac} and f_{coac} are obtained by restricting the domains of f to the accessible and coaccessible states, X_{ac} and X_{coac}, respectively.

Let G_1 and G_2 be two automata. Then $G_1 || G_2$ denotes the parallel composition of G_1 and G_2 (Cassandras and Lafortune, 2008)

The projection operation $P_s : \Sigma^* \to \Sigma_s^*$, where $\Sigma_s \subset \Sigma$ is defined as $P_s(\varepsilon) = \varepsilon, P_s(\sigma) = \sigma$ if $\sigma \in \Sigma_s$ or $P_s(\sigma) = \varepsilon$, if $\sigma \in \Sigma \backslash \Sigma_s$, and $P_s(s\sigma) = P_s(s)P_s(\sigma)$ for all $s \in \Sigma^*$. The projection operation can be applied to a language L by applying $P_s(s)$ to all traces $s \in L$. The inverse projection $P_s^{-1} : \Sigma_s^* \to 2^{\Sigma^*}$ is defined as $P_s^{-1}(t) = \{s \in \Sigma^* : P_s(s) = t\}$, and can also be applied to languages.

The prefix-closure of a language $L \subseteq \Sigma^*$ is given by $\overline{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*) \wedge (st \in L)\}$, and a language is said to be prefix-closed if $L = \overline{L}$. Note that the language generated by an automaton is prefix-closed by definition. A language $L \subseteq \Sigma^*$ is said to be live if for all $s \in L$, there exists $\sigma \in \Sigma$ such that $s\sigma \in L$.

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, for the sake of simplicity, it is assumed that there is only one fault event, *i.e.*, $\Sigma_f = \{\sigma_f\}$.

A fault trace is a sequence of events s such that σ_f is one of the events that form s. A fault-free trace, on the other hand, does not contain event σ_f. The language $L_N \subset L$ denotes the set of all fault-free traces of L, and the subautomaton of G that generates L_N is denoted by G_N. In addition, the set of all fault traces generated by the system is given by $L_F = L \backslash L_N$.

Definition 1. (Sampath et al., 1995) Let L and $L_N \subset L$ be the prefix-closed and live languages generated by G and G_N, respectively, and define the projection operation $P_o : \Sigma^* \to \Sigma_o^*$. Then L is said to be diagnosable with respect to projection P_o and the set of fault events Σ_f if

$$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, ||t|| \geq z) \Rightarrow P_o(st) \notin P_o(L_N),$$

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

In Cabral and Moreira (2020), the definition of synchronous diagnosability of a DES is presented. In order to do so, it is assumed that the system is composed of r modules $G_k = (Q_k, \Sigma_k, f_k, q_{0,k})$, $k = 1, \ldots, r$, *i.e.*, the composed plant is given by $G = ||_{k=1}^r G_k$. It is also assumed that the event set of each module G_k can be partitioned as $\Sigma_k = \Sigma_{k,o} \dot{\cup} \Sigma_{k,uo}$, where $\Sigma_{k,o}$ and $\Sigma_{k,uo}$ denote the sets of observable and unobservable events of G_k, respectively. In this scheme, if an event is observable for one module G_i, and is defined for another module G_j, then it is observable for G_j. In addition, each component has its fault-free behavior modeled by automaton $G_{N_k} = (X_{N_k}, \Sigma_k \setminus \Sigma_f, f_{N_k}, x_{0,k})$.

In the synchronous diagnosis strategy, a diagnoser D_k is constructed for each module, performing the online state estimation of each fault-free model G_{N_k}. If an event $\sigma \in \Sigma_{k,o}$ generated by the plant is observed by D_k, and σ is feasible in at least one state of the current state estimate of G_{N_k}, then the state estimate is updated. Otherwise, if σ is not feasible for all states of the current state estimate of G_{N_k}, then D_k indicates that the fault has occurred. In this scheme, the diagnosers D_k, $k = 1, \ldots, r$, do not communicate with each other, and run in parallel to perform diagnosis. This leads to the following definition of synchronous diagnosability.

Definition 2. (Synchronous diagnosability). Let $G_N = \|_{k=1}^{r} G_{N_k}$, and let L_{N_k} denotes the language generated by G_{N_k}, for $k = 1, \ldots, r$. Let $P_o : \Sigma^\star \rightarrow \Sigma_o^\star$, with $\Sigma_o = \cup_{k=1}^{r} \Sigma_{k,o}$. Then, L is synchronously diagnosable with respect to P_o, L_{N_k}, $k = 1, \ldots, r$, and Σ_f if

$$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq z) \Rightarrow P_o(st) \notin L_{N_a},$$

with $L_{N_a} = \cap_{k=1}^{r} P_{k,o}^{o^{-1}}(P_{k,o}^{k}(L_{N_k}))$, where $P_{k,o}^{o} : \Sigma_o^\star \rightarrow \Sigma_{k,o}^\star$, and $P_{k,o}^{k} : \Sigma_k^\star \rightarrow \Sigma_{k,o}^\star$ are projections. □

According to Definition 2, the system language L is synchronously diagnosable if any occurrence of the fault event σ_f can be detected after a number $z \in \mathbb{N}$ of event occurrences after the fault, by at least one diagnoser D_k constructed based on module G_{N_k}.

3. COMPUTATION OF ALL MINIMAL SUBSETS OF MODULES FOR SYNCHRONOUS DIAGNOSIS

The main advantage of the synchronous diagnosis approach is that the size of the diagnoser grows polynomially with the number of system modules, which reduces the memory space required to store the diagnoser in comparison with traditional techniques. This reduction can be in some cases even greater since, depending on the fault and on the system model, the modules needed for diagnosis can be a subset of the complete set of system modules. In this section, we present a method to compute all minimal subsets that ensure the synchronous diagnosability of the system language.

Let $I_r = \{1, 2, \ldots, r\}$ denote the index set of all system modules. Thus, our objective is to find all minimal subsets $S \in 2^{I_r}$, such that L is synchronously diagnosable with respect to P_o, L_{N_k}, for $k \in S$, and Σ_f, *i.e.*, we want to find the minimal sets S such that

$$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq z) \Rightarrow P_o(st) \notin L_{N_a},$$

with $L_{N_a} = \cap_{k \in S} P_{k,o}^{o^{-1}}(P_{k,o}^{k}(L_{N_k}))$. It is important to remark, as it can be straightforwardly deduced from Definition 2, that if the system language L is synchronously diagnosable with respect to S, then it is synchronously diagnosable with respect to any subset $S' \in 2^{I_r}$ such that $S \subset S'$, *i.e.*, the monotonicity property is valid. Thus, if we obtain all minimal subsets $S \in I_r$ that ensure synchronous diagnosability, then we are able to provide all possible subsets of modules that ensure synchronous diagnosability.

Definition 3. The subset of indexes $S \in 2^{I_r}$ such that the associated modules ensure the synchronous diagnosability of the system language with respect to P_o, L_{N_k}, for $k \in S$, and Σ_f, is called a Synchronous Diagnosis Module Basis (SDMB). □

In order to compute all minimal SDMB, we first present the method proposed in Cabral and Moreira (2020) for the verification of synchronous diagnosability. In the first step of the method presented in Cabral and Moreira (2020), automaton G_F, whose generated language is $\overline{L_F}$, is computed by following three steps (Moreira et al., 2011): 1) label with F all states of G that are reached after the fault event; 2) mark the states labeled with F; and 3) take the coaccessible part of the resulting automaton. Then, automata $\tilde{G}_{N_k}$ is computed from automata G_{N_k} by renaming its unobservable events, transforming them into private events of $\tilde{G}_{N_k}$. Finally, the verifier is computed as $G_V = (\|_{k=1}^{r} \tilde{G}_{N_k}) \| G_F$. The system language L is not synchronously diagnosable with respect to P_o, L_{N_k}, for $k \in I_r$, and Σ_f, if and only if, G_V has a cyclic path cl such that the coordinates associated with G_F of the states of cl have label F, and at least one of the events of the cyclic path cl belongs to Σ.

A method to verify if S forms a SDMB can be obtained by constructing the verifier restricted to the modules associated with S, $G_V^S = (\|_{k \in S} \tilde{G}_{N_k}) \| G_F$, and then searching for the existence of a cyclic path in G_V^S that violates the synchronous diagnosability condition. Thus, if an exhaustive search method is used to find all minimal SDMB, then it is necessary to compute $2^r - 1$ verifiers G_V^S, for each non-empty subset $S \in 2^{I_r}$. In order to mitigate the exponential complexity problem, we present in Algorithm 1, a method to compute all minimal SDMB for the system language.

In Step 1 of Algorithm 1, M and N, representing the set of all minimal SDMB and the set of module indexes k such that $\{k\}$ is not a minimal SDMB, respectively, are defined as the empty set. In Step 2, for each module k, the verifier $G_V^{\{k\}}$ is computed as the parallel composition of the renamed fault-free behavior model $\tilde{G}_{N_k}$ and the fault automaton G_F. Then, the indexes of the verifiers that do not have cyclic paths that violate the synchronous diagnosability condition are added to M as minimal SDMB, and those that have this kind of cyclic path are added to N. In Step 3, the set S_{G_V} of verifiers $G_V^{\{k\}}$, such that $k \in N$, is created. This step is important due to the fact that the indexes $k \in M$ already form minimal SDMB with cardinality one, and therefore, are not added to other subsets when searching for minimal SDMB with cardinality greater than one. In Step 4 the recursive inclusion of modules using Algorithm 2 is performed. Finally, in Step 5 the SDMB that are not minimal are removed from M.

Algorithm 2 is responsible for the recursive module inclusions to form the SDMBs. In Step 1, a path p of verifier G_V^S, associated with a fault sequence that violates the synchronous diagnosability, is obtained, and in Step 2, the subautomaton $G^S_{V_p}$ formed from path p is computed. In Step 3 the loop to add modules is started, and the addition of modules to S is carried out each module at a time, as described in the following steps. Firstly, in Step 3.1, if there does not exist an element of M, E, such that $E \subseteq S \cup \{j\}$, the automaton $G_V^{S\cup\{j\}}$ is computed as the parallel composition of the verifier of the j-th module, that is being checked if it can be added to S, $G_V^{\{j\}}$, and the partial verifier, $G^S_{V_p}$. If there exists a cyclic path that violates the synchronous diagnosability in $G_V^{S\cup\{j\}}$, it means that module j is not capable of eliminating the violating cyclic path associated with p, and consequently, it will not be added to S. Otherwise, if no violating cyclic path remains in the test performed in Step 3.1.1, j is added to S in Step 3.1.2 and verifier $G_V^{S\cup\{j\}} = G_V^S \| G_V^{\{j\}}$ is computed. In step 3.1.2.1, it is verified if $S \cup \{j\}$ forms a SDMB. If $S \cup \{j\}$ forms a SDMB, then it is added to M, else, a new module is searched to be added to $S \cup \{j\}$ by running function ADD_MODULE($G_V^{S\cup\{j\}}$,S_{G_V}).

Algorithm 1. Computation of all minimal SDMB

```
Input: $\tilde{G}_{N_k}$, for $k \in I_r$, and $G_F$
Output: Set of all minimal SDMB, $M$.
1: $M \leftarrow \emptyset$, $N \leftarrow \emptyset$
2: For $k \in I_r$
   2.1: Compute $G_V^{\{k\}} = \tilde{G}_{N_k} \| G_F$
   2.2: If $L$ is synchronously diagnosable with respect to $P_o$, $L_{N_k}$, and $\Sigma_f$, then $M \leftarrow M \cup \{\{k\}\}$, else $N \leftarrow N \cup \{k\}$
3: $S_{G_V} = \{G_V^{\{k\}} : k \in N\}$
4: For $k \in N$
   4.1: ADD_MODULE($G_V^{\{k\}}$, $S_{G_V}$)
5: Find all elements of $M$, $E$, such that there exists another element $E'$, where $E \subset E'$, and eliminate $E'$ from $M$
```

Algorithm 2. ADD_MODULE(G_V^S,S_{G_V})

```
1: Find a path $p$ of $G_V^S$ that departs from its initial state with an embedded cyclic path $cl$ that violates the synchronous diagnosability
2: Compute the subautomaton of $G_V^S$ from path $p$, denoted as $G^S_{V_p}$
3: For $j \in N \setminus S$
   3.1: If there does not exist $E \in M$ such that $E \subseteq S \cup \{j\}$ then
      3.1.1: Compute $G^{j,S}_{V_p} = G_V^{\{j\}} \| G^S_{V_p}$
      3.1.2: If there does not exist a cyclic path in $G^{j,S}_{V_p}$ associated with the cyclic path $cl$ of $G_V^S$ then compute $G_V^{S\cup\{j\}} = G_V^S \| G_V^{\{j\}}$
         3.1.2.1: If there does not exist a cyclic path violating the synchronous diagnosability in $G_V^{S\cup\{j\}}$, then $M \leftarrow M \cup \{S \cup \{j\}\}$, else
            3.1.2.1.1: ADD_MODULE($G_V^{S\cup\{j\}}$,$S_{G_V}$)
```

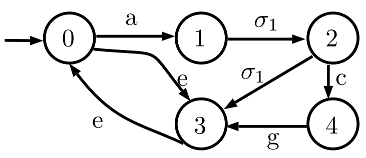

Figure 1. Automaton G_1.

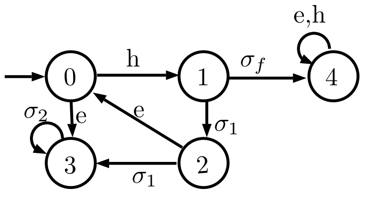

Figure 2. Automaton G_2.

Theorem 1. Algorithm 1 computes all minimal synchronous diagnosis module basis considering fault-free behavior models of the system modules given by G_{N_k}, $k \in I_r$, and fault behavior model given by G_F.

Proof: Note that in Algorithm 1, each element of M is computed recursively by adding to it only modules that eliminates a cyclic path violating the synchronous diagnosability, until there does not exist anymore this kind of cyclic path. Thus, all elements of M are SDMB. In Step 5 of Algorithm 1, all elements of M that contains another element of M are removed, and consequently, all redundant SDMB are eliminated from M. In addition, note that Algorithm 1 computes all possible subsets of I_r by adding to S, incrementally, each module of I_r that eliminates a cyclic path that violates the synchronous diagnosability. Thus, the elements of M forms all minimal SDMB of the system. □

Remark 1. An algorithm is proposed in Santoro et al. (2017) for the computation of all minimal sets of observable events that guarantee the system diagnosability. The main difference in comparison with the method proposed in this paper is that we are searching for sets of modules, and not sets of events, that ensure the synchronous diagnosability of the system language. □

It is important to remark that the minimum SDMB can be obtained by searching in the set of all minimal SDMB, M, those that have the smallest cardinality.

4. EXAMPLE

Let us consider the system composed of the modules G_1, G_2, G_3, and G_4, depicted in Figures 1, 2, 3 and 4, respectively. In order to compute the verifiers $G_V^{\{k\}}$, for $k = 1, 2, 3, 4$, it is first needed to obtain the automata that represent the fault-free languages of each module, using the method proposed in Cabral and Moreira (2020), presented in Figures 5, 6, 7 and 8, respectively, and the automaton that represents the fault language G_F in Figure 11.

In this section we compare the computational cost of obtaining all minimal SDMB using the exhaustive search method, with the method proposed in this paper.

Performing the exhaustive search to find all minimal SDMB, two subsets are obtained, namely the subsets

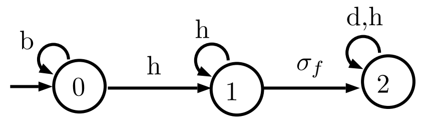

Figure 3. Automaton G_3.

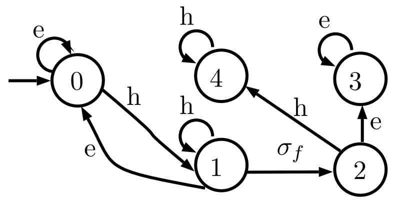

Figure 4. Automaton G_4.

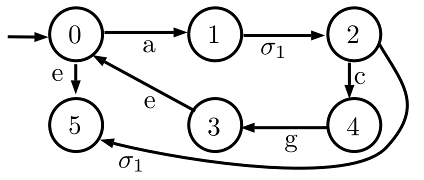

Figure 5. Automaton G_{N_1}.

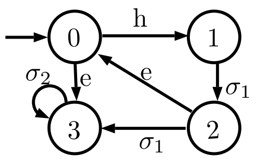

Figure 6. Automaton G_{N_2}.

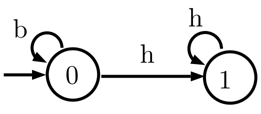

Figure 7. Automaton G_{N_3}.

formed of modules $\{2,3\}$ and $\{3,4\}$. In this procedure, 13 automata must be computed, which results in the computation of 368 states and 794 transitions. The time spent in the process was 1,341.17ms, and the complete information about the verifiers that are computed using the exhaustive search is presented in Table 1.

Using the method proposed in this paper, following the steps of Algorithm 1, instead of computing several parallel compositions with verifiers with a large number of states, partial verifiers $G^S_{V_p}$, presented in Table 2 are calculated, as those depicted in Figures 9 and 10. Since all verifiers G^S_V, such that S is a singleton, have a cyclic path that violates the synchronous diagnosability, then Algorithm

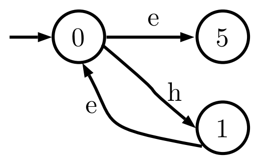

Figure 8. Automaton G_{N_4}.

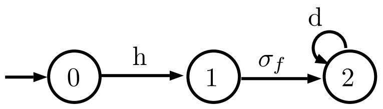

Figure 9. Automaton $G^{\{1\}}_{V_p}$ formed from the cyclic path of $G^{\{1\}}_V$ that violates the synchronous diagnosability of the system language considering only module $\{1\}$.

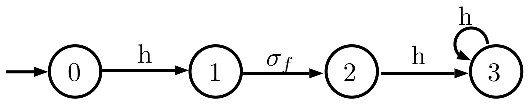

Figure 10. Automaton $G^{\{3\}}_{V_p}$ formed from the cyclic path of $G^{\{1\}}_V$ that violates the synchronous diagnosability of the system language considering only module $\{3\}$.

2 is recursively repeated until all minimal SDMB are computed. In this procedure, the verifiers presented in Table 1 are computed. Note that verifiers $G^{\{1,4\}}_V$, $G^{\{2,4\}}_V$, and $G^{\{1,3,4\}}_V$, that are computed in the exhaustive search method, are not computed using the proposed method.

The total number of automata that are computed using Algorithm 1, presented in Tables 1, 2, and 3, was 24, with total number of states equal to 318 and with total number of transitions 613. The time spent in the process of obtaining all minimal SDMB, $\{2,3\}$ and $\{3,4\}$, using Algorithm 1, was 922.50ms. This shows a reduction of 31% in time, 14% in the number of states and 23% in the number of transitions, as shown in Table 4, in comparison with the exhaustive search method. Knowing all minimal SDMB it is possible to obtain all minimum SDMB, and in this case, $\{2,3\}$ and $\{3,4\}$.

5. CONCLUSIONS

In this paper, we propose an algorithm to find all minimal SDMB of the language of a DES. The computational cost of the proposed algorithm is compared with the exhaustive search method, and we show that, with the proposed method, there is a significant reduction in the number of states and transitions of the automata needed to compute the minimal SDMB for the proposed example. We are currently investigating strategies to reduce even more the computational cost of the method in order to mitigate the exponential complexity of computing all minimal SDMB.

REFERENCES

Basile, F., Cabasino, M.P., and Seatzu, C. (2017). Diagnosability analysis of labeled time petri net systems. *IEEE Transactions on Automatic Control*, 62(3), 1384–1396.

Basile, F., Chiacchio, P., and de Tommasi, G. (2009). An effcient approach for online diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, 54(4), 748–759.

Cabasino, M.P., Giua, A., and Seatzu, C. (2014). Diagnosability of discrete-event systems using labeled petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1), 144–153.

Cabral, F.G. and Moreira, M.V. (2020). Synchronous diagnosis of discrete-event systems. *IEEE Transactions*

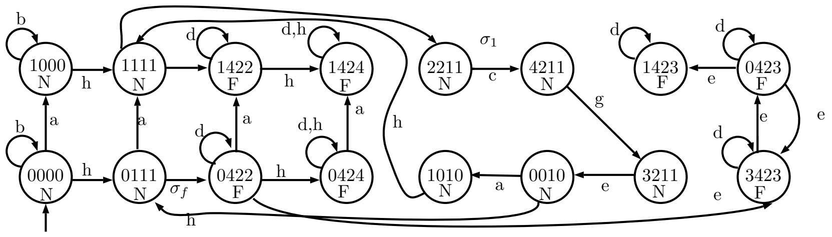

Figure 11. Automaton G_F.

Table 1. Number of states and transitions of the verifiers G_V^S computed using the exhaustive search and the proposed method.

Method	S	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$	$\{1,2,3\}$	$\{1,2,4\}$	$\{1,3,4\}$
Exhaustive	States	26	28	16	14	48	26	22	28	28	14	48	48	22
search	Transitions	57	65	25	24	123	48	42	56	65	19	110	123	37
Proposed	States	26	28	16	14	48	26	-	28	-	14	48	-	22
method	Transitions	57	65	25	24	123	48	-	56	-	19	110	-	37

Table 2. Number of states and transitions of the partial verifiers $G_{V_p}^S$ that are computed in the example using Algorithm 1.

$G_{V_p}^S$	$G_{V_p}^{\{1\}}$	$G_{V_p}^{\{2\}}$	$G_{V_p}^{\{3\}}$	$G_{V_p}^{\{1,2\}}$	$G_{V_p}^{\{1,3\}}$
States	3	3	4	3	4
Transitions	3	3	4	3	4

Table 3. Number of states and transitions of the testing automata $G_{V_p}^{j,S}$ computed using Algorithm 1.

$G_{V_p}^{j,S}$	$G_{V_p}^{2,\{1\}}$	$G_{V_p}^{3,\{1\}}$	$G_{V_p}^{4,\{1\}}$	$G_{V_p}^{3,\{2\}}$	$G_{V_p}^{4,\{2\}}$	$G_{V_p}^{4,\{3\}}$	$G_{V_p}^{3,\{1,2\}}$	$G_{V_p}^{4,\{1,2\}}$	$G_{V_p}^{4,\{1,3\}}$
States	7	3	3	3	3	3	3	3	3
Transitions	13	2	3	2	3	2	2	3	2

Table 4. Total number of states and transitions that are computed using the exhaustive search and the proposed method, and reduction in the computational cost and execution time.

	States	Transitions	Execution Time (ms)
Exhaustive search	368	794	1,341.17ms
Proposed method	318	613	922.50ms
Reduction	14%	23%	31%

on Automation Science and Engineering, 17(2), 921–932.

Cabral, F.G., Moreira, M.V., Diene, O., and 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.

Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer-Verlag New York Inc., Secaucus, NJ, 2 edition.

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

Debouk, R., Malik, R., and Brandin, B. (2002). A modular architecture for diagnosis of discrete event systems. In *41st IEEE Conference on Decision and Control*, 417–422. Las Vegas, Nevada USA.

Fanti, M.P., Mangini, A.M., and Ukovich, W. (2013). Fault detection by labeled petri nets in centralized and distributed approaches. *IEEE Transactions on Automation Science and Engineering*, 10(2), 392–404.

Gascard, E. and Simeu-Abazi, Z. (2013). Modular modeling for the diagnostic of complex discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 10(4), 1101–1123.

Moreira, M.V., Jesus, T.C., and Basilio, J.C. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 56(7), 1679–1684.

Sampath, D., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 40(9), 1555–1575.

Sampath, D., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1996). Failure diagnosis using discrete-event models. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, 4(2), 105–124.

Santoro, L.P.M., Moreira, M.V., and Basilio, J.C. (2017). Computation of minimal diagnosis bases of discrete-event systems using verifiers. *AUTOMATICA*, 77, 93–102.

Zad, S.H., Kwong, R., and Wonham, W. (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 48(7), 1199–1212.