Diagnosis of Discrete Event Systems by Independent Windows

Xingyu Su[1,2] and **Alban Grastien**[1,2]
[1]Optimisation Research Group, NICTA, Australia
[2]Artificial Intelligence Group, Australian National University, Australia
e-mail: u4383016@anu.edu.au, alban.grastien@nicta.com.au

Abstract

We propose new algorithms for diagnosis of discrete event systems that slice the observations into windows, each diagnosed independently. Doing so allows to cope with intermittent observations and to ignore the overhead of maintaining a precise estimate of the system state. We show how diagnosability can be asserted using this approach – and any diagnosis algorithm in general – through the notion of a *simulation*, which is a modified model that simulates how the diagnosis algorithm computes the diagnosis.

1 Introduction

Diagnosis of discrete event systems (DES) is performed by computing the paths on the complete model that generate the observations received on the system, or equivalently the belief state of system. This belief state can be computed on-line, by iteratively computing the set of states that can be reached from the current belief state through transitions that would produce exactly the next observation. If this is done explicitly [Baroni *et al.*, 1999], the number of these states makes the approach inapplicable for many real-world problems. Symbolic approaches have been proposed where the states are represented in propositional logic, e.g., Binary Decision Diagram (BDD), but this representation is also subject to exponential blow-up. Finally, the potential belief states can be pre-computed [Sampath *et al.*, 1995], but their number is double exponential this time.

In this last approach, the exponential blow-up stirs from the fact that the diagnoser maintains all the information it can about the observations it received so far. For instance, the current belief state could be $\mathcal{B}_1$ instead of $\mathcal{B}_2 \simeq \mathcal{B}_1$ because of an early observation.

In this paper, we propose several algorithms for computing diagnosis of DES, which share the property that they only consider the most recent observations. On top of the computational advantages mentioned above, these algorithms are able to deal with intermittent loss of communication, whereby the state of the system becomes unknown.

On the other hand, independent windows may potentially lead to imprecise diagnosis. In order to measure the quality of this approach, we adapt the well-known criterion of diagnosability. Diagnosability is the property of an observable system that states that, using the model, a fault can be diagnosed after it occurs. We extend this definition to algorithms, and say that an algorithm is diagnosable if it will diagnose the fault after it occurs. We show that diagnosability can be tested using known algorithms, on the condition that simulation can be built, i.e., a modified model that *simulates* the diagnostic algorithm decision.

This paper is organised as follows. Section 2 provides the traditional definitions of DES model and diagnosis. Section 3 defines the generic notion of diagnostic algorithm and the issue of proving that an algorithm is diagnosable. Section 4 presents the independent-window algorithms and how diagnosability can be tested for such algorithms. Section 5 demonstrates an example on how to implement diagnosability testing.

2 Diagnosis and Preliminaries

2.1 Diagnosis of DES

We choose DES as a modelling framework [Cassandras and Lafortune, 2008]. A DES models dynamic systems at a discrete level. We use automata to represent DES, although we also give definitions at the language level. Faults can be defined at the event level or the state level; to simplify the following definitions, we consider faulty states.

Definition 1 (Automaton) *An* automaton *is a tuple $\langle Q, \Sigma, T, I, L\rangle$ where Q is a finite set of* states, *Σ is a finite set of* events, *$T \subseteq Q \times \Sigma \times Q$ is a set of* transitions, *$I \subseteq Q$ is the set of* initial states, *and $L : Q \to \{N, F\}$ is a* mode label function *where N stands for* nominal *and F for* faulty *such that $(\langle q, e, q'\rangle \in T \wedge L(q) = F) \Rightarrow L(q') = F$.*

The system is modelled by an automaton with a set $\Sigma_o \subseteq \Sigma$ of observable events. For ease, we can also see the model as a pair of languages $\mathcal{L} = \mathcal{L}_N \cup \mathcal{L}_F$ such that $e_1, \ldots, e_k \in \mathcal{L}_l$ ($l \in \{N, F\}$) iff there exists a trace $q_0 \xrightarrow{e_1} \ldots \xrightarrow{e_k} q_k$ where $q_0 \in I$ and $L(q_k) = l$.

The observation $obs(\sigma)$ of $\sigma \in \Sigma\star$ is defined as the restriction of σ to the set of observable events:

$$obs(e_1, \ldots, e_k) = \begin{cases} \varepsilon & \text{if } k = 0 \\ e_1, obs(e_2, \ldots, e_k) & \text{if } e_1 \in \Sigma_o \\ obs(e_2, \ldots, e_k) & \text{if } e_1 \in \Sigma \setminus \Sigma_o \end{cases}$$

Fig. 1 shows a graphical representation for a DES model. It visualises a DES model and its states, initial

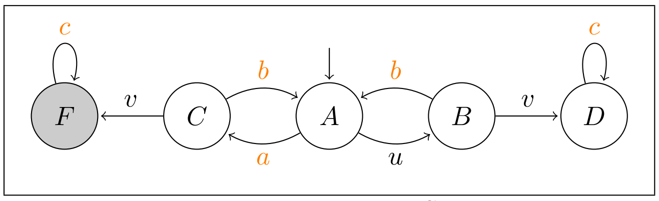


Figure 1: Example DES model

state, transitions, events and the results of the mode label function L. Finally, observable events are a, b, c, with the rest being unobservable, u and v.

A diagnosis problem is therefore defined as a model and an observation. In general, the diagnosis should indicate whether the system is in nominal mode, in faulty mode, or whether it cannot decide, i.e., ambiguous state. In this work, the diagnosis policy does not distinguish between nominal mode and ambiguous mode, i.e., the diagnoser assumes that the system is not faulty unless proved otherwise.

Definition 2 (Diagnosis) *The* diagnosis $\Delta(M, o)$ *of observation o using model M is defined by:*

- *N if $\exists \sigma \in \mathcal{L}_N.\ obs(\sigma) = o$ and*
- *F otherwise.*

Most existing algorithms implement a similar definition [Sampath *et al.*, 1995; Baroni *et al.*, 1999; Pencolé and Cordier, 2005; Su and Wonham, 2005; Grastien *et al.*, 2007; Kan John and Grastien, 2008]. Because her work is the original one, we shall call Δ the *Sampath* diagnosis.

Observation	Diagnoser	Diagnosis
a, b, b, b, b, c, c, c	Sampath	N
a, b, b, b, a, c, c, c	Sampath	F
b, a, b, a, c, c, c, c	Sampath	F

Table 1: Example of diagnosis results

Example Tab. 1 shows some examples of diagnosis results for the system on Figure 1. For instance, given a sequence of observations a, b, b, b, b, c, c, c, the Sampath Diagnoser will return a nominal diagnosis.

2.2 Diagnosability of DES

Diagnosability is the question whether a fault will always be diagnosed. This property is very desirable and the system designer might want to enforce it.

Diagnosability can be checked by searching for faulty system behaviours that cannot be diagnosed precisely (in dynamic systems where the aim is to diagnose the fault eventually, these system behaviours must be infinite). Any such example is a proof that the system is not diagnosable; it is called an ambiguous trace. Failure to find such a trace proves that the system is diagnosable, assuming the search was complete. In this reduction of the diagnosability problem, the model is used twice [Grastien and Torta, 2011]: it is used i) to find the faulty trace on the system and ii) to (unsuccessfully) diagnose the trace. However we may have to use different models for these two usages, one model for the system and the other model for the diagnosis. If the system model is abstracted before being used for diagnosis, the abstracted model should not be used to generate the faulty trace (that trace may not be a possible system behaviour); on the other hand, whether the trace can be diagnosed should be tested with the diagnosis model.

Definition 3 (Diagnosability of a model) *A diagnosis model M is* diagnosable *w.r.t a system model M' if the following holds true:*

$$\exists n \in \mathbf{N}.\ \forall s \in \mathcal{L}'_F.\ \forall t \in \Sigma\star\,.$$
$$(st \in \mathcal{L}' \wedge |t| \geq n \Rightarrow \Delta(M, obs(st)) = F)\,.$$

Given a faulty behaviour s, given an additional n events (represented by t), the diagnosis (using the model M) of the observation $obs(st)$ of st should be "faulty".

Jiang et al. [2001] have shown that diagnosability can be checked in polynomial time with respect to the number of states. To this end they build a structure called a *twin plant*, which is the classical automata synchronisation of two copies of the model on the observable events. M is diagnosable w.r.t M' iff the twin plant contains no infinite cycle of states $\langle q, q' \rangle$ where $L(q) = N$ and $L(q') = F$.

3 Diagnostic Algorithms

In this section we discuss the notion of diagnostic algorithm which is a possibly imprecise implementation of the diagnosis presented in Definition 2.

Definition 4 (Diagnostic Algorithm) *A* diagnostic algorithm *is a function A :* MODELS $\times\ \Sigma_o\star \to \{N, F\}$ *where* MODELS *is the set of possible models and the following holds:*

Monotonicity:
$A(M, o) = F \Rightarrow \forall e \in \Sigma_o.\ A(M, o\ e) = F$; and

Correctness: *$A(M, o) = F \Rightarrow \Delta(M, o) = F$.*

The first condition ensures that the diagnosis is monotonic [Lamperti and Zanella, 2007], i.e., that if a fault has been diagnosed, this conclusion will not be withdrawn. The second condition of the definition ensures that the diagnosis is correct, i.e., that the algorithm returns "faulty" only when the system is faulty (the converse may not hold).

The diagnosis problem is a complex task. We are interested in considering algorithms that may return results that are less precise than the Sampath diagnosis, but that run faster. We also want however to be able to give guarantees about the output of the algorithm, e.g., diagnosability.

Definition 5 (Diagnosability of an Algorithm) *The diagnostic algorithm A is* diagnosable *for a diagnosis model M w.r.t. a system model M' if the following holds true:*

$$\exists n \in \mathbf{N}.\ \forall s \in \mathcal{L}'_F.\ \forall t \in \Sigma\star\,.$$
$$(st \in \mathcal{L}' \wedge |t| \geq n \Rightarrow A(M, obs(st)) = F)\,.$$

This definition is very similar to that of diagnosability (Definition 3) except that we require the output of the algorithm A (and not the Sampath diagnosis itself) to be "faulty".

Diagnosability of a model can be tested using the twin plant approach. In general however, we do not know how to compute the diagnosability of a diagnostic algorithm. To this end we propose to define a new model $si(M, A)$ that "simulates" the behaviour of the diagnosis algorithm.

Definition 6 (Simulation) *Given a diagnostic algorithm A and a model M, the* simulation *of A and M is an automaton $si(M, A)$ s.t. $\forall o \in \Sigma_o\star$. $A(M, o) = \Delta(si(M, A), o)$.*

Using the simulation, we can use the following theorem to prove diagnosability.

Theorem 1 *Algorithm A is diagnosable for a diagnosis model M w.r.t. a system model M' iff $si(M, A)$ is diagnosable w.r.t. M' where $si(M, A)$ is the simulation of M and A.*

Proof of Theorem 1 Based on definition 5 for diagnosability and the condition that diagnosis of an algorithm A is equivalent to that of Sampath Diagnoser using a simulated model, $si(M, A)$, the diagnosability of Sampath Diagnoser using $si(M, A)$ implies the diagnosis on a trace st should return F if it reaches faulty state, i.e., $\Delta(obs(st)) = F$.

Hence, the diagnosis of the algorithm A on the trace st should also return F if it reaches faulty state. $A(M, obs(st)) = F$.

Therefore, it is proved that the diagnosability of (A, M, M') is equivalent to the diagnosability of $(\Delta, M, si(M, A))$. □

Theorem 1 shows that it is possible to use the twin plant method to decide whether algorithm A is diagnosable for M.

4 Independent-Windows Algorithm

A time window is defined as a slice of a sequence of observations. Independent-windows algorithms perform diagnosis on either a single window or a collection of time windows. In this section, we explain the motivations for the independent-windows algorithms, four different independent-window algorithms, and their associated simulation.

4.1 Motivations

There are essentially three main motivations for independent-windows algorithms: the flexibility of this approach, its potential impact on the diagnosis complexity, and its use to tackle the problem of masking.

The independent-windows algorithms perform independent diagnosis analyses on separate windows. Because each diagnosis is performed separately, we can skip the ones that we believe are not relevant. Consider for instance a problem occurring on a web-service whose activity is logged. It might be tedious to go through all the logs to produce the diagnosis; it is more likely that analysing the data some time before the incident was detected will be enough.

Second, we expect the complexity of diagnosis to become more manageable by considering independent-windows. Diagnosis of DES as defined above is about maintaining a belief state, and the number of such belief states is exponential in the number of system states (double exponential in the number of variables). Whichever way the belief state is computed (off-line [Sampath *et al.*, 1995] or on-line [Baroni *et al.*, 1999], even using decomposition or symbolic tools), diagnosis quickly becomes untrackable. With windows of fixed size however, the complexity changes completely. For instance, given b observable events in the model and a size k of the window, the number of Sampath belief states that can be reached is $\Sigma_{i\in\{0,\dots,k\}} b^i = \frac{b^{k+1}-1}{b-1} = O(b^k)$. Practically, this means that a Sampath diagnoser of polynomial size can be built, or the BDD unfolding on k transitions of the system model.

Finally when the communication layer used to transmit the observation is also subject to faults, certain observations are *masked* which means that the complete sequence of observations is not available. Independent-windows algorithms are less subject to this issue since they reset the diagnosis at the beginning of every window.

4.2 Single-Window Diagnosis

Single-window diagnosis computes the diagnosis on windows, which are small chunks of observations. Given a sequence $o = o_1, \dots, o_n \dots$ of observations, given two indexes $i < j$, the window $o[i, j]$ is defined as the subsequence of observations $o_i, \dots, o_j$. Notice that, if $j \le n$, the length of the window (number of observations) is $j - i + 1$; if $i < n$, its size is 0; otherwise, the size is $n - i + 1$.

Given a window $o_1, \dots, o_k$, a system behaviour $\sigma \in \mathcal{L}$ is compatible with the window if the window appears in the observations of σ; formally if there exist two sequences of observations $o'_1, \dots, o'_m$ and $o''_1, \dots, o''_p$ such that $obs(\sigma) = o'_1, \dots, o'_m, o_1, \dots, o_k, o''_1, \dots, o''_p$.

Definition 7 (Single-Window Algorithm) *The* single-window algorithm *denoted $W_{i,j}$ is defined as follows:*

- *$W_{i,j}(M, o) = N$ if there exists a nominal behaviour $\sigma \in \mathcal{L}_N$ compatible with $o[i, j]$;*
- *$W_{i,j}(M, o) = F$ otherwise.*

Observe that the diagnosis $W_{i,j}(M, o)$ can be simply computed as the diagnosis $\Delta(M', o[i, j])$ where M' is the modified model M where all reachable states are made initial.

4.3 Windows-Based Diagnosis

Based on single-window diagnosis, windows-based diagnosis facilitates various ways to slice a sequence of observations into time windows, each of which comes with its strengths. The final output will be a conjunction of diagnostic results of several time windows, i.e., the system is in faulty mode if the diagnosis on at least one time window returns faulty; it is in nominal mode if all time windows return nominal.

Given an observation $o \in {\Sigma_o}^\star$, given the sub-words $o_1, \dots, o_n$ of o and Time-Windows $= \{[i_1, j_1], \dots\}$ where i_i and j_i are integer indexes as defined in Section 4.2, the following diagnosis is correct:

$$A(M, o) = \begin{cases} N & \text{if } \forall [i, j] \in \text{Time-Windows}. \\ & \quad W_{i,j}(M, o) = N \\ F & \quad \text{otherwise.} \end{cases}$$

Notice that more observations implies that the observations of the windows increase, which increases the chances for a single-window diagnosis to return faulty, hence the monotonicity of the algorithm.

The definition in Section 4.3 facilitates future time windows. If the system has been diagnosed faulty, then the diagnosis of any following time windows will return faulty. Therefore, windows-based diagnosis obeys the properties of monotonicity and correctness as defined in Definition 4. In the next section, we will illustrate windows-based diagnosis by four examples. In Section 4.5, we will examine the diagnosability of windows-based diagnosis.

4.4 Examples of Windows-Based Diagnosis

We will provide descriptions for the time windows of four independent-windows algorithms and their expected benefits. These algorithms should then follow the definitions in Section 4.2 and 4.3.

Algorithm 1 (Al_1) *Al_1 means slicing a sequence of observations every k observations:*
Time-Windows-$Al_1 = \{[1, k], [k + 1, 2k], \dots\}$.

Algorithm 2 (Al_2) *Al_2 makes sure the windows overlap so that observations which are consecutive appear in the same window:*
Time-Windows-$Al_1 = \{[1, k], [k + 1, 2k], \dots\} \cup \{[k' + 1, k' + k], [k' + k + 1, k' + 2k], \dots\}$ where $k' = \lfloor \frac{k}{2} \rfloor$.

Algorithm 3 (Al_3) *Al_3 aims at slicing observations and running diagnosis at random times, yet frequently reoccurring, times:*
Time-Windows-$Al_3 = \{[i_1, i_1 + k], [i_2, i_2 + k], \dots\}$ such that there exists a maximum delay d, and for all index x, $0 < i_x - i_{x-1} < d$.

Algorithm 4 (Al_4) *Al_4 proposes sliding time windows to run diagnosis. A sliding time window moves along a observation flow by one observation at a time.*
Time-Windows-$Al_4 = \{[1, k], [2, k + 1], \dots\}$.

Observation	Slice	Diagnosis	Output
a, b, b, b, b, c, c, c	(a, b, b, b)	N	N
	(b, c, c, c)	N	
a, b, b, b, a, c, c, c	(a, b, b, b)	N	F
	(a, c, c, c)	F	
b, a, b, a, c, c, c, c	(b, a, b, a)	N	N
	(c, c, c, c)	N	

Table 2: Examples of Al_1

Example Tab. 2 shows the results of Al_1 running on the same set of observations as in Tab. 1. Based on the DES model in Fig. 1, given the first input: a, b, b, b, b, c, c, c, using Al_1 with the window size to be 4, we get 2 slices: (a, b, b, b) and (b, c, c, c). Running Al_1 on the first slice leads to N. Also, the diagnostic result of the second slice is N. Therefore, this sequence of observations is N. One of the slices of the second observation ends in F. Thus, Al_1 will return F. In the third observation, the diagnostic results of both slices are N. However, Tab. 1 shows this observation should be F. This demonstrates that Al_1 has its drawbacks of imprecise diagnosis in some situations. Algorithm Al_2 overcomes this issue.

Input	Slice	Diagnosis	Output
a, b, b, b, b, c, c, c	(a, b, b, b)	N	N
	(b, b, b, c)	N	
	(b, c, c, c)	N	
a, b, b, b, a, c, c, c	(a, b, b, b)	N	F
	(b, b, a, c)	F	
	(a, c, c, c)	F	
b, a, b, a, c, c, c, c	(b, a, b, a)	N	F
	(b, a, c, c)	F	
	(c, c, c, c)	N	

Table 3: Examples of Al_2

Tab. 3 is the result of running Al_2, which uses additional slices. We have seen that Al_1 cannot diagnose the fault in the third observation. In contrast, Al_2 will accurately return F.

4.5 Simulation of the system model M'

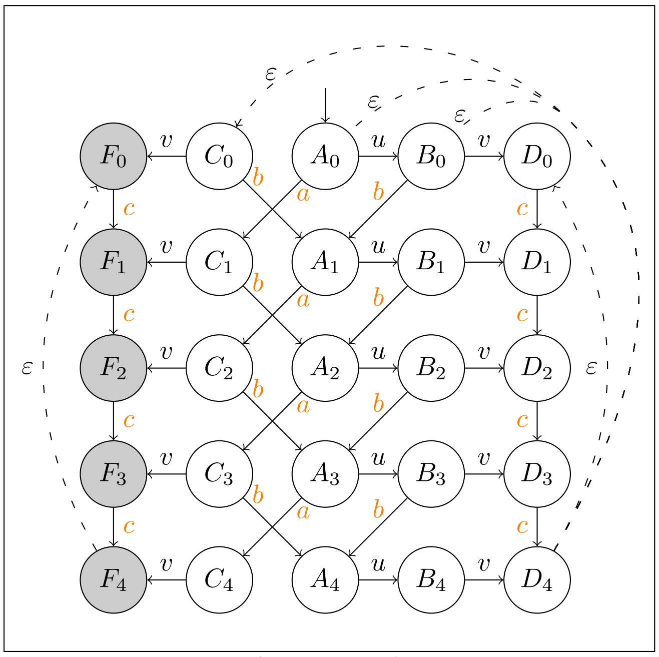
We now study diagnosability in order to evaluate windows-based algorithms by presenting a simulation of the window-based algorithms. Remember that, while the simulation generates a model with more states than the original one, the simulation is only used for diagnosability testing. Furthermore, the states and transitions of the automata do not need to be explicitly represented, but can be represented symbolically, e.g. BDD.

The simulation $si(M, Al_1)$ of the window-based algorithm Al_1 and the model M is an automaton that contains $k + 1$ copies of the states of M, where each state $\langle q, i\rangle$ of $f(M, Al_1)$ records not only the system state q but also the number i of observations in the current window. When the number of observations has reached k, then the state of the simulation is "reset", i.e., a transition takes the simulation from state $\langle q, k\rangle$ to state $\langle q', 0\rangle$ such that q' and q only have in common that their diagnostic information is the same: $L(q) = L(q')$.

Definition 8 *The k-simulation of $M = \langle Q, \Sigma, T, I, L\rangle$ is the automaton $M' = \langle Q', \Sigma', T', I', L'\rangle$ where $Q' = Q \times \{0, \dots, k\}$, $\Sigma' = \Sigma \cup \{\varepsilon\}$, $I' = I \times \{0\}$, $L'(\langle q, i\rangle) = L(q)$, and $T' = T_u \cup T_o \cup T_\varepsilon$ defined by:*

- $T_u = \{\langle\langle q, i\rangle, u, \langle q', i\rangle\rangle \mid i \in \{0, \dots, k\} \wedge \langle q, u, q'\rangle \in T \wedge u \in \Sigma \setminus \Sigma_o\}$,
- $T_o = \{\langle\langle q, i\rangle, o, \langle q', i + 1\rangle\rangle \mid i \in \{0, \dots, k - 1\} \wedge \langle q, o, q'\rangle \in T \wedge o \in \Sigma_o\}$, *and*
- $T_\varepsilon = \{\langle\langle q, k\rangle, \varepsilon, \langle q', 0\rangle\rangle \mid L(q) = L(q')\}$.

Example Fig. 2 illustrates the Algorithm Al_1 simulation for the DES model in Fig. 1. F_0, F_1, F_2, F_3 and F_4 are faulty states, with the rest being nominal. Definition 8 says for every observable transition, create a link from one state to the next state on the next row,



ε
ε
ε
ε
ε
F_0 C_0 A_0 B_0 D_0
F_1 C_1 A_1 B_1 D_1
F_2 C_2 A_2 B_2 D_2
F_3 C_3 A_3 B_3 D_3
F_4 C_4 A_4 B_4 D_4
v
u
a
b
c

Figure 2: Part of Algorithm Al_1 simulation for the DES model in Fig. 1. Dotted lines also need to link A_4 to A_0, B_0, C_0 and D_0. Same applies to B_4 and C_4.

which represents the next time step; for every unobservable transition, create a link from one state to the next state within its row; for each faulty state at the end of one time window, create a link ε from F_4 to every faulty state at the beginning of one time window, i.e., F_0; for each nominal state at the end of the time window, create a link ε to every nominal state at the beginning of the time window.

There is one unique simulation that can be built for a system model M' and an algorithm A. Simulation is only used for diagnosability test and not for diagnosis. In the simulation of Al_2, Fig. 1 and 2 are synchronised on the initial states, i.e., A_0 and A_2, each leading to a simulation. For Al_3, every time window has a simulation, which may have different sizes as determined by the time windows of Al_3. Finally, those simulations will be synchronised on their initial states to reach the simulation of Al_3.

With the help k-simulation, we develop Theorem 2.

Theorem 2 *The k-simulation of a diagnosis model M as defined in Definition 8 is the simulation of Al_1 and M as defined in Definition 6.*

Proof of Theorem 2 By observing that a path on the k-simulation of M with $|o|$ observations is the concatenation of $\lceil |o|/k \rceil$ paths of length k such that the diagnostic information at the end of one such path is the same as the one at the end of the other path. Therefore all paths in the simulation consistent with the observations end in a faulty state iff all paths of one window end in a faulty state, which is the definition of the simulation of a diagnosis algorithm. □

The simulations, together with Theorem 1, allow to prove that Algorithm Al_1 with $k \geq 2$ is not diagnosable for the system of Figure 1; on the other hand, Algorithm Al_2 is diagnosable for any $k \geq 2$.

5 Experiments

We adopt BDD as the data structure for DES. We implement BDD using JDD, a Java library for creation and operations on BDD variables[1]. First of all, the input DES model is in *des_comp* format according to the Dia-Des project of Yannick Pencolé[2]. Second, we use BDD variables for every state, event, and then build transitions. Third, we synchronise multiple automata on shared events. Next, we build a twin plant and implement the Forward Algorithm of symbolic model checking in order to test diagnosability [Grastien, 2009]. This is the first approach to test diagnosability without introducing any windows-based algorithm. In the second approach, before building a twin plant, make a copy of the synchronised automaton, unfold that copy in order to get the simulation for the windows-based algorithm, and then build a twin plant to test diagnosability. All data and benchmark are available on request.

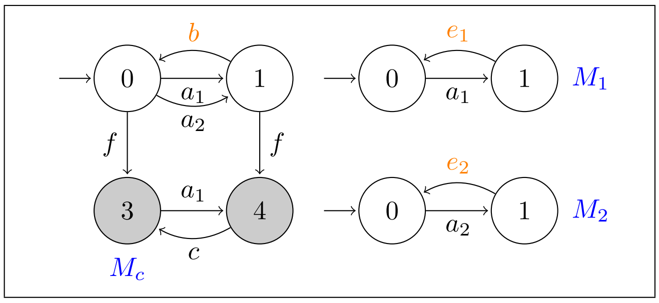


Figure 3: Experiments.

In order to test diagnosability when the windows-based algorithm is introduced, we build an example of factory operations, which consists of a few DES models. Fig. 3 shows a central model, M_c, and multiple operation plants, i.e., M_1, M_2, etc. M_c is to dispatch a job (a_i) to each plant (M_i) and receive feedback (e_i) once it is done. States 3 and 4 are shaded as faulty, with the rest being nominal. In the individual operation plants, e_i will be emitted. However, b will be emitted from M_c. Although c is not observable from M_c, only e_1 will be observed from M_1 if M_c enters a faulty scenario. To sum up, b and e_i are observable, while a_i, f and c are unobservable.

The experiments were run on a Ubuntu 12.04 computer with Intel Core i7-3610QM processor @3.3GHz and the total memory is 8 GB.

The results show that they are all diagnosable. Fig. 4 is a line chart to show log_{10} of the running time of diagnosability tests in milliseconds. The X-axis lists the number of model components to be synchronised. We start from 2 components, M_c and M_1, and then synchronise with M_2, M_3, etc. The Y-axis shows the time consumption in milliseconds. In Fig. 4, the filled boxes represent the time of running diagnosability test without any windows-based algorithm. The filled diamonds show the time when including the unfolding operation on the simulation of the windows-based algorithm as defined in Definition 8.

[1]javaddlib.sourceforge.net/jdd/
[2]homepages.laas.fr/ypencole/diades/html/index.html

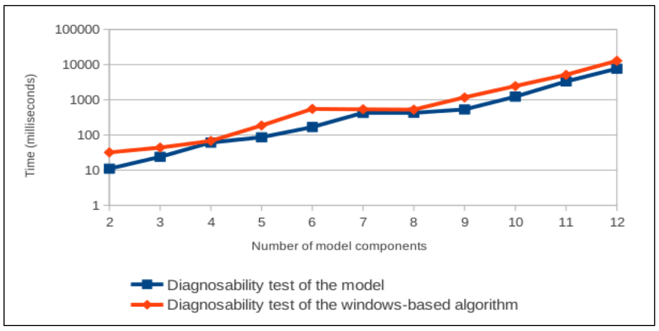


Figure 4: Running time

In summary, the trend of time consumption meets our expectation as the amount of model components grows. Also, diagnosability test on the windows-based algorithm takes more computation time.

6 Conclusion

We develop Independent-Windows Algorithms to diagnose DES and we study their diagnosability. We aim to slice a sequence of observations into time windows and diagnose them independently. This approach has the advantage to handle intermittent observations and does not require maintaining a precise estimate of the system state.

We begin with studying three key properties of a general diagnosis algorithm, such as monotonicity, correctness and diagnosability. We then prove Theorem 1 to determine the diagnosability of any diagnosis algorithm using a modified model and the twin plant method. This theorem enables us to compute the diagnosability of the new Independent-Windows Algorithms, so that we can measure how precise the diagnosis will be.

As we are certain that we will be able to determine the diagnosability of our new diagnosis algorithms, we are further motivated by the flexibility of independent windows, reduction of computation complexity and avoidance of the masking issues which may occur in transmission of observations. We then demonstrate four Independent-Windows Algorithms. Using the same set of observations, we compare the accuracy of Sampath Diagnoser and the new algorithms Al_1 and Al_2. This allows us to analyse the strengths of each diagnosis algorithms.

In order to determine the diagnosability of the Independent-Windows Algorithms as specified in Theorem 1, we show how to modify a DES model and simulate a diagnosis algorithm. After this, we use the output simulation to test the diagnosability of the new algorithms. Theorem 2 proves the correctness of generating the simulation. Finally, our experiments conclude that diagnosability testing on the windows-based algorithms can be implemented by BDD.

As an outline for our future work, we plan to study how prompt the windows-based diagnosis is. We will measure this by how fast it can reach a diagnostic result. We also want to examine the root cause of ambiguity if the diagnosis result is inability to decide whether the system is nominal or faulty. One possibility is to identify key states in the simulation, categorise them and retain this information.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

The authors want to thank Yannick Pencolé and the three anonymous reviewers for their useful comments.

References

[Baroni *et al.*, 1999] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence (AIJ)*, 110(1):135–183, 1999.

[Cassandras and Lafortune, 2008] Ch. Cassandras and S. Lafortune. *Introduction to discrete event systems (2nd ed.)*. Kluwer Academic Publishers, 2008.

[Grastien and Torta, 2011] A. Grastien and G. Torta. A theory of abstraction for diagnosis of dicrete-event systems. In *Ninth Symposium on Abstraction, Reformulation and Approximation (SARA-11)*, pages 50–57, 2011.

[Grastien *et al.*, 2007] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *22nd Conference on Artificial Intelligence (AAAI-07)*, pages 305–310, 2007.

[Grastien, 2009] A. Grastien. Symbolic testing of diagnosability. In *20th International Workshop on Principles of Diagnosis (DX-09)*, pages 131–138, 2009.

[Jiang *et al.*, 2001] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control (TAC)*, 46(8):1318–1321, 2001.

[Kan John and Grastien, 2008] P. Kan John and A. Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *Eighteenth European Conference on Artificial Intelligence (ECAI-08)*, 2008.

[Lamperti and Zanella, 2007] G. Lamperti and M. Zanella. On monotonic monitoring of discrete-event systems. In *Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, pages 130–137, 2007.

[Pencolé and Cordier, 2005] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)*, 164(1–2):121–170, 2005.

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

[Su and Wonham, 2005] R. Su and W. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE Transactions on Automatic Control (TAC)*, 50(12):1923–1935, 2005.