Window-based Diagnostic Algorithms for Discrete Event Systems: What Information to Remember

Xingyu Su[1,2] and **Alban Grastien**[1,2] and **Yannick Pencolé**[3,4]
[1]Optimisation Research Group, NICTA, Australia
[2]Artificial Intelligence Group, Australian National University, Australia
[3]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[4]Univ de Toulouse, LAAS, F-31400 Toulouse, France
e-mail: u4383016@anu.edu.au, alban.grastien@nicta.com.au, ypencole@laas.fr

Abstract

Window-based diagnostic algorithms only use the most recent observations to diagnose a discrete event system. This approach can cause precision loss if less recent observations are necessary to understand the system behavior. This paper formally presents two extensions of window-based algorithms that solve this issue by carrying over some information about the current state of the system from one window to the next. We then show how precision for the two algorithms can be decided using our precision test depending on the amount of information that is retained. Finally, we propose a procedure that minimizes this amount of information so that the proposed algorithms remain precise.

1 Introduction

This paper addresses the problem of on-line diagnosis of discrete event systems (DES) as initially proposed in [1]. Given a flow of observable events generated by the underlying system the problem consists in determining whether the DES is normally operating or not based on a behavioral model of it. Many previous works address this problem by the use of a diagnoser automaton [1; 2], automata unfoldings [3] that can also be distributed as in [4; 5] or encoded with binary decision diagrams (BDD) as in [6], and finally by the use of a satisfiability solver as in [7]. The main challenge is to deal with the complexity of the diagnosis algorithm that has to monitor on the fly the observable flow and generate a succession of belief states consistent with the flow: the difficulty is the size of each belief state that is exponential in the number of states of the system. Any of the precited work propose diagnosis algorithms that attempt to compute at any time a belief state that is consistent with the observable flow from the time the system starts operating till the current time. The main drawback of such a conservative strategy is the inability to *follow* the observable flow for large system due to the exponential size of the generated belief states and therefore the temporal complexity to handle them.

Another strategy is to give way of the conservative strategy by proposing diagnosis algorithms that only apply on the very last events of the observable flow and *forget* about the past: this is the *window-based approach* as introduced in our previous work [8]. Obviously, this approach suffers from precision loss: both current and past observations may be necessary to understand the system behavior and this precision loss can be measured with the simulation technique described in [9][1].

In this paper, we propose to compromise between these two extreme strategies by looking at the minimum piece of information to *remember* from the past (abstracted belief state) so that a window-based algorithm will certainly ensure the same precision as any conservative algorithms. Our first contribution is to formally present two window-based algorithms extending an algorithm from [8] that carry over some information about the current state of the system from one window to the next. Our second contribution is to describe how the precision of these new algorithms can be verified with respect to the simulation method [9]. Finally, we propose a formal procedure to minimize the amount of information that a window-based algorithm needs to carry over to ensure no precision loss.

This paper is organized as follows. Section 2 recalls the necessary background about the diagnosis problem and reviews the existing window-based algorithms, called Independent-Window Algorithms (IWAs). Section 3 presents the new algorithms, called Time-Window Algorithms (TWAs). Section 4 demonstrates how to test the precision of each TWA. Section 5 discusses our implementation choices and the procedure that minimizes the amount of information required for a TWA to be precise. Section 6 concludes this study and outlines the future work.

2 Background

2.1 Diagnosis of DES

This work is based on the model-based diagnosis framework of DES [10]. A DES is represented with an automaton that is a tuple $M = \langle Q, \Sigma, T, I, L\rangle$ 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*. The function $L : Q \to \{N, F\}$ assigns to each state a *mode* that is either *nominal* (N) or *faulty* (F) and $L^{-1} : \mathbb{L} \to Q$ denotes its reverse function. We will assume in this paper that once the system is faulty, it remains faulty: $(\langle q, e, q'\rangle \in T \wedge L(q) = F) \Rightarrow L(q') = F$. A *trace* is any sequence of transitions $q_0 \xrightarrow{e_1} \dots \xrightarrow{e_k} q_k$ of the automaton M. It is also denoted $q_0 \xrightarrow{\sigma} q_k$ where $\sigma = e_1 \dots e_k$ is the sequence of events. Any system behavior is represented by a trace such that $q_0 \in I$.

[1]This in-press paper is available at:
http://www.grastien.net/ban/articles/sg-ecai14.pdf

The model has a set of observable events $\Sigma_o \subseteq \Sigma$ and a set of unobservable events $\Sigma \setminus \Sigma_o$. Any sequence σ has an observable projection $obs(\sigma)$ recursively defined as follows: $obs(\varepsilon) = \varepsilon$ (empty sequence), $obs(\sigma e) = obs(\sigma)e$ if $e \in \Sigma_o$, $obs(\sigma e) = obs(\sigma)$ if $e \notin \Sigma_o$; finally $obs(S)$ denotes the set of observable projections of the sequences in S.

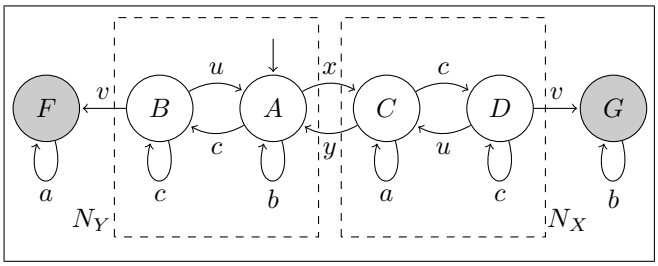

Figure 1: DES model: a, b, c, x, y are observable events and u, v are unobservable events.

Figure 1 depicts a DES model. State A is the unique initial state. States F and G are faulty and the other states are nominal.

The diagnosis problem is defined based on the notion of *consistent trace*.

Definition 1 (Consistent Trace)**.** *The trace $q \xrightarrow{\sigma} q'$ is* consistent *with a sequence of observations θ if $obs(\sigma) = \theta$.*

Let $Q_0, \ldots, Q_p \subseteq Q$ be a collection of $p+1$ subsets of states, let $\theta_1, \ldots, \theta_p$ be a collection of p sequences of observations,

Definition 2 (ct predicate)**.** *The predicate $ct([Q_0, \ldots, Q_p], [\theta_1, \ldots, \theta_p])$ holds iff there exists a trace $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_p} q_p$ such that $q_0 \in Q_0$, and for any $i \in \{1, \ldots, p\}$, $q_i \in Q_i$, $\sigma_i \in \Sigma^*$, and the trace $q_{i-1} \xrightarrow{\sigma_i} q_i$ is consistent with θ_i.*

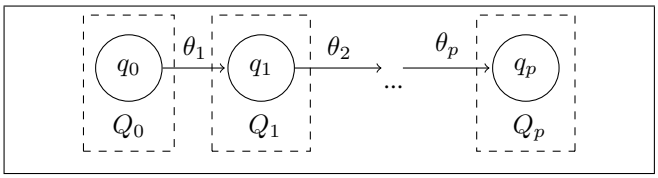

Figure 2: Visual representation for the ct predicate

Figure 2 is a visual representation for the ct predicate. The visual representations for two TWAs in Section 3 are built on Figure 2.

Throughout this paper, we will assume that the system is diagnosable [1]: for any trace $q_0 \xrightarrow{\sigma} q_{k-1} \xrightarrow{e} q_k$ with $q_0 \in I, \sigma \in \Sigma^*, L(q_{k-1}) = N, L(q_k) = F$, there always exists a number n such that for any trace $q_0 \xrightarrow{\sigma} q_{k-1} \xrightarrow{e} q_k \xrightarrow{\sigma'} q_l, \sigma' \in \Sigma^*$, as soon as $(|obs(\sigma')| \geq n)$ the set of traces $q'_0 \xrightarrow{\sigma''} q'_m, q'_0 \in I, \sigma'' \in \Sigma^*$ consistent with $obs(\sigma.e.\sigma')$ are such that $L(q'_m) = F$. Based on the diagnosability assumption, we can adopt an optimistic view for diagnosing the system: as long as at least one nominal explanation of the observations can be found, the diagnosis will assume the system as nominal. Indeed, if the system is faulty, diagnosability ensures that a finite number of future observations is required to assert that no nominal explanation exists any more: the diagnosis will then certainly assert that the system is faulty.

Definition 3 (Diagnosis Δ)**.** *Given a model M and a sequence of observations Θ, the* diagnosis $\Delta(M, \Theta)$ *is N if $ct([I, L^{-1}(N)], [\Theta])$ holds, and F otherwise.*

2.2 Diagnostic Algorithms and Precision

As stated in the introduction, we propose a set of diagnosis algorithms. Let MODELS be the set of models M as introduced in the previous section. A *diagnostic algorithm* is a function $A : \text{MODELS} \times (\Sigma_o^*) \to \{N, F\}$ such that the following conditions hold:

Correctness: $A(M, \Theta) = F \Rightarrow \Delta(M, \Theta) = F$;

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

The first condition 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 second condition ensures that the diagnosis is monotonic [11], i.e., that if a fault has been diagnosed, this conclusion will not be withdrawn by additional observed events.

The previous definition of a diagnostic algorithm does not give any constraint on the precision of the diagnosis. The important criterion is whether the algorithm is able to return "faulty" (F) after a fault effectively occurred on the system, possibly after some delay: in this case, the diagnostic algorithm is said to be *precise*: if $\Delta(M, \Theta) = F$ then for any possible and sufficiently long continuation Θ' such that $A(M, \Theta\Theta') = F$.

2.3 Independent-Window Algorithms (IWAs)

The algorithms presented in this paper are based on the Independent-Windows Algorithms (IWAs) algorithms that we recently proposed [8]. IWAs slice the flow of observations $\Theta = o_1, \ldots, o_n$ into small, independent windows θ_i that are diagnosed independently. IWAs aim at improving flexibility (because windows are diagnosed separately and in parallel) and reducing complexity (because the size of the windows is bounded). However because the links between the windows are lost, the precision of the IWAs may be reduced; this happens for instance when a fault can be diagnosed only by observing two specific events that never appear in the same window.

A window θ is a sub-sequence $o_i o_{i+1} \ldots o_j$ (also denoted $\Theta[i, j]$) of the actual observations Θ. The diagnosis of a window θ consists in determining whether there exists a nominal trace that generates this sub-sequence of observations. IWAs diagnose each window separately, and return the system is faulty as soon as the fault is diagnosed for one window.

We have proposed four IWAs (namely Al_1,...,Al_4) [8] that differ only in the window selection. The existing IWAs diagnose k observations for one time window, and move to another time window without keeping any information. The simplest variant, Al_1, slices the sequence of observations into consecutive windows of identical length. Al_2 ensures that the time windows overlap so that any consecutive observations o_i, o_{i+1} always appear at least in one selected time window. Al_3 constructs "sample" time windows, i.e., such that not every observation appears in a window. Al_4 diagnoses on sliding time windows, i.e., the first window is $\Theta[1, k]$, the second $\Theta[2, k+1]$, etc. A sliding window moves by one observed event at a time.

3 Time-Window Algorithms (TWAs)

We propose two Time-Window Algorithms (namely Al_5 and Al_6) that extend the previous IWAs. The basic principle of these time-window algorithms is to remember from one time window to the next some of the knowledge about the current diagnosed state. Al_5 remembers the key information from the previous consecutive time window whereas Al_6 remembers the history from the previous overlapping time window.

3.1 Motivation Example

Information propagated by the proposed TWAs throughout the time windows relies on the notion of *abstract state* of the system. Going back to Figure 1, N_X and N_Y are such abstract states. The current abstract state is N_Y if the current state is within $\{A, B\}$ and the abstract state is N_X if the current state is within $\{C, D\}$. Note that these abstract states have been chosen so that there are discriminable (just after the observation of x (resp. y), the system is definitely in N_X (resp. N_Y)), although they need not be in general. Moreover, in this example, if the system is in N_X, b is a symptom of a fault but not a, and if the system is in N_Y, a is a symptom of a fault but not b. Faults can thus be precisely diagnosed simply by keeping track of the current abstract state and checking for each a and each b whether they are not symptomatic from this current abstract state.

On the one hand, none of the four IWAs is precise for this example. This is because IWAs only look at bounded groups of consecutive observations, while the distance between the last abstract state change and the first observation showing the failure may be arbitrarily large (the observations in between may be all c).

On the other hand, it is unnecessary to remember precisely the current state of the system. The only relevant piece of information about the current state of the system is the abstract state of the system. This is precisely what the two algorithms that we introduce here do: we assume a rough partition of the state space (how this partition is chosen is the topic of Section 5), and the information that is passed from one window to the next is the subset of abstract states the current system state may belong to.

3.2 Time-Window Algorithm Al_5

Al_5 relies on a partition $\Pi = \{Q_1, \ldots, Q_z, Q_F\}$ of the state-space Q where Q_F denotes the faulty states. Let $\mathbb{L} = \{N_1, N_2, \ldots, N_z, F\}$ be a set of labels such that any N_i represents the nominal states Q_i and F represents the faulty states Q_F.

Definition 4 (Abstract State)**.** *An* abstract state *is an element Q_i of the state-space partition Π and is represented by his corresponding label of $\mathbb{L}$.*

Al_5 slices a sequence of observations every k observations (as the IWA Al_1 does) while remembering some information from the previous time window, by computing an *abstract belief state* as defined below.

Definition 5 (Abstract Belief State)**.** *An* abstract belief state *is any non-empty subset of $\mathbb{L}$.*

There are two major differences between Al_1 and Al_5. First, Al_5 has several *labels* of nominal states, i.e., $\mathbb{L}_5 = \{N_1, N_2, \ldots, N_z, F\}$, while Al_1 only has one *label* of nominal state, i.e., $\mathbb{L}_1 = \{N, F\}$. Second, Al_1 diagnoses k observations for one time window, and moves to the next consecutive time window without keeping any information. For Al_5, diagnosis on one time window considers the abstract knowledge of the system states from the previous time window. The only exception is that the diagnosis for first time window begins with the initial state.

Definition 6 (Al_5)**.** *Given a model M, the initial states I, a state abstraction $\mathbb{L}$, a time window size k, a sequence of observations $\Theta = \theta_1, \theta_2, \ldots, \theta_t$, and the number of time windows $t = \frac{|\Theta|}{k}$,*

$$Al_5(M, \Theta) = \begin{cases} N & \text{if } \exists \ell_1, \ell_2, \ldots, \ell_t \in \mathbb{L} \setminus \{F\} \text{ s.t.} \\ & ct([I, L^{-1}(\ell_1)], [\theta_1]) \wedge \\ & ct([L^{-1}(\ell_1), L^{-1}(\ell_2)], [\theta_2]) \wedge \cdots \wedge \\ & ct([L^{-1}(\ell_{t-1}), L^{-1}(\ell_t)], [\theta_t]) \\ F & \text{otherwise} \end{cases}$$

The observation sequence Θ is sliced into t time windows, and each time window contains a sub-sequence of observations θ_i; these traces must agree on the label at the beginning and the end of the windows. Al_5 returns nominal if for each time window, there exists a trace that is consistent with the sub-sequence of observations, and ends in a nominal state. Otherwise, it returns faulty. Each trace provides an explanation for one time window.

Property 1. *Al_5 is correct and monotonic.*

Proof outline: By construction, Al_5 concludes N as long as it computes a non-empty superset of the nominal traces that are consistent with Θ, hence the correctness. Once Al_5 concludes F, it means there is no more nominal traces consistent with Θ. Monotonicity therefore follows from diagnosability. □

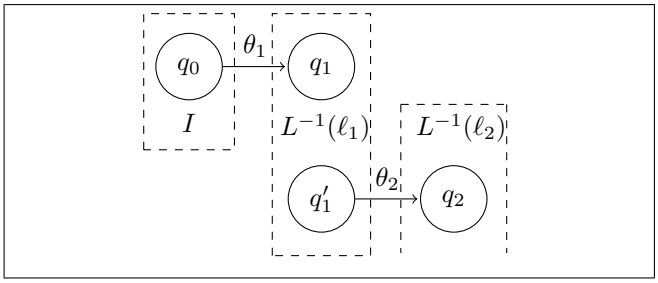

Figure 3: Visual representation for Al_5: The first row is the first time window, and the second row is the next consecutive time window. $L^{-1}(\ell_2)$ will link to the third time window.

Figure 3 is a visual representation for Al_5. I represents the abstract state at the beginning of the first time window, to which the initial states belong, and $L^{-1}(\ell_1)$ represents the abstract state at the end of the first time window. $\theta_1 = obs(\sigma_1)$ is the observations of the trace σ_1 in the first time window, and $\theta_2 = obs(\sigma_2)$ is the observations of the trace σ_2 in the second time window. $[I, L^{-1}(\ell_1)]$ and $[\theta_1]$ represents the diagnosis on the first time window. $[L^{-1}(\ell_1), L^{-1}(\ell_2)]$ and $[\theta_2]$ represents the diagnosis on the next time window. In general, $q_1 \neq q_1'$; however, $L(q_1) = L(q_1')$.

3.3 Example of Al_5

Algorithm	Slice	Abstract Belief State	Diagnosis
Δ	Θ	NA	F
Al_1	bx	$\{N\}$	N
	cc	$\{N\}$	
	cb	$\{N\}$	
Al_5	bx	$\{N_X\}$	F
	cc	$\{N_X\}$	
	cb	$\emptyset$	

Table 1: Diagnostic results of Δ, Al_1 and Al_5 for DES in Figure 1 and observations $\Theta = bxcccb$. To simplify the notation, Definition 6 excludes any fault. $\emptyset$ means there is no nominal abstract belief state and thus it is faulty.

Table 1 shows the diagnostic results of observations $\Theta = bxcccb$ when $k = 2$ for the DES model in Figure 1. The abstract state labels of Al_5 are $\mathbb{L} = \{N_X, N_Y, F\}$. For any time window θ_i, Al_1 concludes N (there always exists a nominal trace σ_i starting from a nominal state of the system such that $obs(\sigma_i) = \theta_i$) whereas Al_5 is able to diagnose this sequence precisely (there is no nominal trace σ_3 starting from a state belonging to N_X such that $obs(\sigma_3) = \theta_3 = cb$).

3.4 Discussion of Al_5

First, Al_5 is more precise than Al_1, but this comes at a price. Al_5 does not take the advantage of parallel computation and must analyze time windows in order while Al_1 can diagnose multiple time windows independently and simultaneously.

Furthermore, Al_5 is more vulnerable to the *masking* issue than Al_1. Observations are *masked* when the communication layer used to transmit the observations is momentarily faulty. Even if the complete sequence of observations is not available, Al_1 resets the diagnosis at the beginning of every time window and therefore Al_1 is immune to the masking issue (that is, if the fault does not occur during the masking). In comparison, when certain observations are masked and a fault actually occurs, Al_5 relies on the assumption that the abstract state can be estimated with sufficient precision, but the masking can wipe out this estimation.

3.5 Motivation for Time-Window Algorithm Al_6

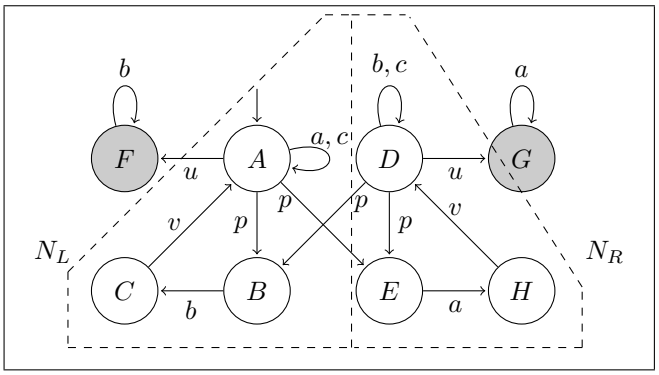

Figure 4: DES model with a set of abstract states $\{N_L, N_R\}$ where Al_5 is not precise. Only u and v are unobservable.

We now present a more sophisticated example on Figure 4 that motivates Al_6. Suppose that the selected partition splits the nominal states into two abstract states $N_L = \{A, B, C\}$ and $N_R = \{D, E, H\}$. In this setting, a consecutive sequence of b's (resp. a's) means the system is faulty if the system is known to be in N_L (resp. N_R).

Diagnoser	Slice	Abstract Belief State	Diagnosis
Δ	Θ	NA	F
Al_5	ap	$\{N_L, N_R\}$	N
	ac	$\{N_L, N_R\}$	
	ca	$\{N_L\}$	
	aa	$\{N_L\}$	

Table 2: Diagnostic results of Δ and Al_5 for DES in Figure 4: observations $\Theta = apaccaaa$

Table 2 shows the diagnostic results of Al_5 when $k = 2$.

If a sequence of observations $\Theta = apaccaaa$ is sliced to ap and $accaaa$, the abstract belief state immediately after a, p should be N_R. However, Al_5 is unable to precisely determine the abstract belief state after a, p, which makes it unable to diagnose the fault. This is because this model requires two observable events to recognize the transition to a different abstract state, i.e., pa means transition to N_R and pb means transition to N_L.

3.6 Time-Window Algorithm Al_6

Compared to Al_5, Al_6 slices a sequence of observations every k observations and includes additional overlapping time windows. Assume k is a multiple of 2 and let $h = \frac{k}{2}$, the time windows selected by Al_6 are $\{w\_6(1) = [1, 2h], w\_6(2) = [h + 1, 3h], w\_6(3) = [2h + 1, 4h], w\_6(4) = [3h + 1, 5h], \dots\}$. Al_6 also refines the carry-over information of any time window i (denoted $@i$) and passes it to the next overlapping time window $i + 1$ (denoted $@i + 1$), so that inconsistent abstract states will be eliminated.

Definition 7 (Al_6)**.** *Given a model M, the initial states I, a state abstraction $\mathbb{L}$, a time windows size k with $h = \frac{k}{2}$, a sequence of observations Θ, and the number of time windows $t = \frac{|\Theta|}{h} - 1$, $Al_6(M, \Theta)$ returns N if*

$$\begin{aligned}
&\exists \ell_0@i, \ell_1@i, \ell_2@i \in \mathbb{L} \setminus \{F\}\ \forall i \in \{1, \dots, t\}\ \textit{s.t.}\\
&\quad ct([I, L^{-1}(\ell_1@1), L^{-1}(\ell_2@1)], [w_6(1)])\\
&\wedge\ ct([L^{-1}(\ell_0@2), L^{-1}(\ell_1@2), L^{-1}(\ell_2@2)], [w_6(2)])\\
&\wedge\ \dots\\
&\wedge\ ct([L^{-1}(\ell_0@t), L^{-1}(\ell_1@t), L^{-1}(\ell_2@t)], [w_6(t))])\\
&\wedge\ \ell_1@1 = \ell_0@2 \wedge \dots \wedge \ell_1@(t-1) = \ell_0@t,
\end{aligned}$$

where $w\_6(i) = \Theta[H+1, H+h], \Theta[H+(h+1), H+2h]$, $H = i \times h$; and F otherwise.

The function w_6 returns the i-th time window, which consists of two equal-length sub-sequences of observations θ_1 and θ_2. Al_6 returns nominal if for each time window, there exists a trace that is consistent with the sub-sequence of observations, and ends in a nominal state. Otherwise, it returns faulty.

Property 2. *Al_6 is correct and monotonic.*

Proof outline: Same as Al_5. □

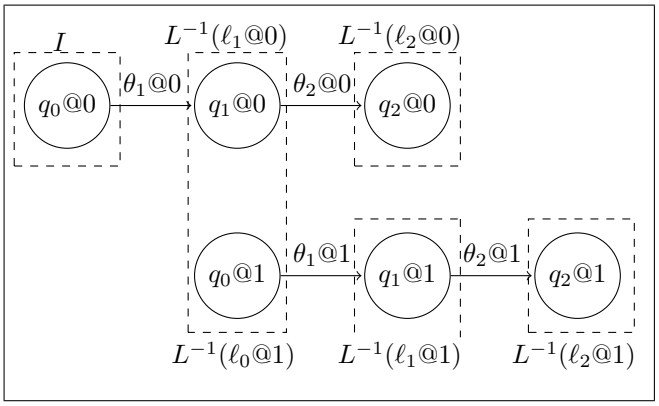

Figure 5: Visual representation for Al_6: The first row is the first time window, and the second row is the next overlapping time window. $L^{-1}(\ell_1@1)$ will link to the third time window.

Figure 5 is a visual representation for Al_6. I represents the abstract state at the beginning of the first time window, to which the initial states belong. The diagnosis of the window j searches a trace that starts from a state of $L^{-1}(\ell_0@j)$, goes through a state of $L^{-1}(\ell_1@j)$ while generating the first half of the observations, and ends in a state of $L^{-1}(\ell_2@j)$ while generating the second half. The rationale behind Al_6 is that the estimation of $\ell_2@j$ is not very precise because it is supported only by previous observations, whilst that of $\ell_1@j$ is supported by both previous and latter observations. Therefore the information that is carried over to the next window is $\ell_1@j = \ell_0@(j+1)$.

3.7 Example of Al_6

Diagnoser	Slice	ABS	CT	Output
Δ	Θ	NA	NA	F
Al_6	ap	$\{N_L\}$	Yes	F
	pa	$\{N_R\}$	Yes	
	ac	$\{N_R\}$	Yes	
	cc	$\{N_R\}$	Yes	
	ca	$\emptyset$	No	
	Algorithm may stop here.			

Table 3: Diagnostic results of Δ and Al_6 for DES in Figure 4 and the same observation as Table 2, i.e., $\Theta = apaccaaa$: ABS means the abstract belief state in the middle of a time window; CT means whether a consistent trace holds.

Table 3 shows the diagnostic results when $k = 2$. Al_6 returns the precise diagnosis using the given abstract states as stated in 3.5. Al_6 is more precise than Al_5, without requiring a more detailed break-down of abstract states. This is because Al_6 refines the diagnosis on the slices. For instance, at the end of the first slice (a, p), the system may be in the state B or E. After the second slice (p, a), Al_6 eliminates the state B because the observation a is impossible from this state. The diagnostic result is more precise because it has been refined to be in the state E only.

4 Precision Test for TWAs

As we have shown, the two algorithms presented here may be unable to detect faults in spite of the diagnosability of the system. It is therefore important to be able to assess whether the imprecision introduced by the algorithms may make some faulty behaviours undetectable.

A diagnostic algorithm Al is precise for a system model M if it always detects and identifies faults. We have shown [8] that this precision test can be implemented by using a *simulation*, which is a finite state machine that models the sources of imprecision of the diagnostic algorithm (cf. the paper for details). Formally, the simulation is a model $si(M, Al)$ such that diagnosing the observations with algorithm Al and model M yields the same results as using algorithm Δ and model $si(Al, M)$:

$$\forall \Theta, \Delta(si(Al, M), \Theta) = Al(M, \Theta).$$

Testing whether any algorithm Al is precise consists in analysing $si(Al, M)$ with a twin-plant method [8]. The simulations for Al_5 and Al_6 can be built accordingly.

Formally, the Al_5-simulation for $M = \langle Q, \Sigma, T, I, L\rangle$ is the automaton $M_5 = \langle Q_5, \Sigma_5, T_5, I_5, L_5\rangle$ where $Q_5 = Q \times \{0, \ldots, k\}$, $\Sigma_5 = \Sigma \cup \{\varepsilon\}$, $I_5 = I \times \{0\}$, $L_5 : Q_5 \to \mathbb{L}$, $L_5(\langle q, i\rangle) = L(q)$, and $T_5 = T_u \cup T_o \cup T_\varepsilon$ defined by:

- $T_u = \{\langle\langle q, i\rangle, u, \langle q', i\rangle\rangle \mid i \in \{0, \ldots, k-1\} \wedge \langle q, u, q'\rangle \in T \wedge u \in \Sigma \setminus \Sigma_o\}$,
- $T_o = \{\langle\langle q, i\rangle, o, \langle q', i+1\rangle\rangle \mid i \in \{0, \ldots, k-1\} \wedge \langle q, o, q'\rangle \in T \wedge o \in \Sigma_o\}$, and
- $T_\varepsilon = \{\langle\langle q, k\rangle, \varepsilon, \langle q', 0\rangle\rangle \mid L_5(q) = L_5(q')\}$.

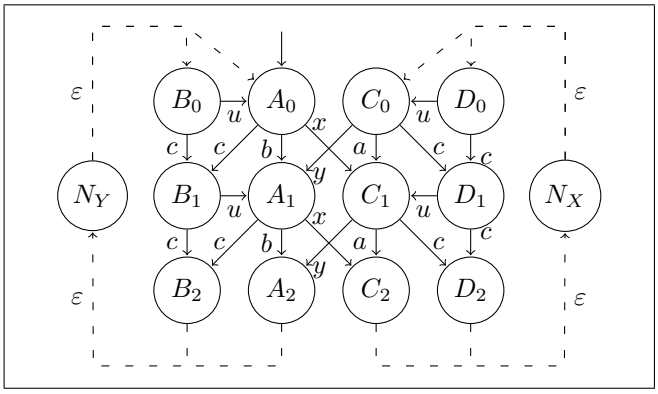

Figure 6: Simulation for Al_5 and the DES model in Figure 1. N_Y and N_X are only used to highlight the abstract states. The faulty states are omitted because the assumptions in Section 2.1 state that we only look for nominal explanations and thus faulty states are not necessary for precision testing.

Figure 6 shows the simulation of Al_5 for the model of Figure 1 with the window size of 2. Each state of the simulation is associated with a counter which simulates the number of observations made in the current window; state A_1 for instance represents the situation where the current system state is A and 1 observation was made so far in the current window. When the counter reaches k (here 2), the end of the window is simulated by an (unobservable) ε-transition that obliterates the current state and only remembers the abstract state (*reset*). For instance, there is an ε-transition from A_2 to B_0 because A and B share the same label N_Y, while there is no transition from A_2 to C_0.

The size of the Al_5-simulation is k times that of the DES model, where k is the time window size. Thus, verifying the precision of Al_5 and a DES model using the Twin Plant

method remains polynomial. Remember that the simulations are only used to verify the precision of the algorithm, and not for diagnosis.

5 Implementation and Optimization

We discuss the implementation of the TWAs and how the knowledge that needs to be carried over can be minimized.

5.1 Implementation

We assume that the system is modeled symbolically: a state is defined as an assignment of Boolean state variables V. A particular subset $V_{\mathbb{L}}$ of these variables V will be used for encoding the information that is carried over the time windows. For instance in Figure 1, the 6 states can be modeled with three variables $V = \{f, p, q\}$. The memory set is $V_{\mathbb{L}} = \{f, p\}$: f evaluates to true when the state is faulty (that is F or G) and p evaluates to true if the state is in N_X or in N_Y. The last variable $q \in V \setminus V_{\mathbb{L}}$ in this example evaluates to true if the last event was c. The state B is represented by formula $\Phi(B) = \neg f \wedge \neg p \wedge q$ and the abstract state $N_Y \in \mathbb{L}$ is encoded with $\Phi(L^{-1}(N_Y)) = \neg f \wedge \neg p$.

Different implementations of the algorithms presented here are possible, using either the diagnoser [1], SAT [7], or BDDs [12]; we concentrate on the latter one. Diagnosis can be performed by tracking the belief state, i.e., the set of states that the system may be in at any time; BDDs allow to manipulate sets of states "efficiently". The implementation of Al_5 is mostly similar to the standard global model approach (see [12]), except that the belief state is updated between two windows: the belief state β' at the beginning of window i is an abstract state that share the same label as one state of the belief state β at the end of window $i-1$: $\beta' = L^{-1}(L(\beta))$ (where L is extended to sets of states). For instance in Figure 1, if $\beta = \{A\}$, then $L(\beta) = \{N_X\}$ and $\beta' = \{A, B\}$.

The remaining question is how to implement the operations L and L^{-1} with BDDs. The symbolic representation allows for a very elegant implementation of L: given a formula $\Phi(C)$ that models a set $C \subseteq \mathbb{L}$ of labels, the formula that models the set of states $L^{-1}(C)$ is the same: $\Phi(L^{-1}(C)) = \Phi(C)$. The L function can be implemented by a logical existential operator: $\Phi(L(Q')) = \exists(V \setminus V_{\mathbb{L}}).\, \Phi(Q')$. In other words, $\Phi(\beta') = \exists(V \setminus V_{\mathbb{L}}).\, \Phi(\beta)$. Indeed $\Phi(\{A, B\}) = \neg f \wedge \neg p = \exists q.\, \Phi(\{B\})$.

5.2 Optimization of a TWA

Finally we are interested in reducing the set of labels, i.e., we want to find a minimal memory set $V_{\mathbb{L}}$. This problem is very interesting because it allows to identify *the minimal amount of information that is needed to summarize the past observations while maintaining precision*. This work relies on the following monotonicity result:

Theorem 1. *If Al_5 or Al_6 is precise for a given model with a memory set $V_{\mathbb{L}}$ then for all other variable $v \in V \setminus V_{\mathbb{L}}$ it is also precise for set $V_{\mathbb{L}} \cup \{v\}$.*

Proof outline: The smaller $V_{\mathbb{L}}$ is, the bigger the superset of nominal consistent traces determined by Al_5 or Al_6 is. If for a given $V_{\mathbb{L}}$, Al_5 or Al_6 concludes F then this superset is empty. So for any $V_{\mathbb{L}} \cup \{v\}$, this superset is also empty, and the precision is kept. □

We propose Procedure 1 to compute a minimal memory set. The procedure is similar to the one used by Brandán-Briones, Lazovik and Dague to find a minimal subset of observable events ensuring diagnosability [13]. Starting with an empty set, variables are added until the algorithm becomes precise. Next all variables are examined to remove the ones that are unnecessary. Notice that the procedure remains polynomial since each loop is applied at most a linear number of times.

Procedure 1 Add and Optimize Variables

```
   Input: TWA, M
   Output: A minimal memory set V_L
1  V_L := {}
2  while TWA is not precise for M do
3  |   w := witness
4  |   Identify v in w
5  |   Add v to V_L
6  foreach v ∈ V_L do
7  |   Remove v from V_L
8  |   if TWA is not precise for M then
9  |   |   Reinsert v to V_L
10 return V_L
```

We explain how to choose the variable to add to $V_{\mathbb{L}}$. The precision algorithm (Line 2) allows to generate a "witness", i.e., a faulty trace of the system together with the nominal trace on the simulation that exonerates the faulty trace; because the system is diagnosable, the simulation trace necessarily contains one (or more) ε-transition as presented in the previous section that changes the value of one (or more) state variable. We choose one of these variables, which will exclude this witness for later precision tests. For instance in Figure 1, states A and C disagree on the value of p; so if the witness contains an ε-transition from A to C, p becomes a candidate variable to add to the memory set.

6 Conclusion

This paper addresses the problem that the existing window-based algorithms remember no information between time windows. We propose the Time-Window Algorithms (TWAs) and present two instances. Although they are incomplete algorithms, we show how the precision of each TWA can be verified by building a Simulation. We also discuss the BDD implementation and propose an procedure that minimizes the amount of information that a TWA needs to carry over. Our contribution is developing from the naive window-based diagnosis to more comprehensive and realistic diagnostic algorithms, which slice a sequence of observations and diagnose the time windows. This approach has the advantage when handling intermittent observations and does not require maintaining a precise estimate of the system state.

We plan to study how to apply window-based diagnosis to non-diagnosable systems since the IWAs and TWAs focus on diagnosable systems so far. We also plan to analyze the probability of the system states when resetting a time window. The goal is to develop more advanced window-based diagnostic algorithms and to further enhance the diagnostic precision.

Acknowledgements

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

The authors also want to thank Patrik Haslum (Artificial Intelligence Group, Australian National University, Australia) and the reviewers for their constructive comments.

References

[1] 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.

[2] L. Rozé and M.-O. Cordier. Diagnosing discrete-event systems : extending the "diagnoser approach" to deal with telecommunication networks. *Journal of Discrete Event Dynamical Systems (JDEDS)*, 12(1):43–81, 2002.

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

[4] 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.

[5] 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.

[6] A. Schumann, Y. Pencolé, and S. Thiébaux. A decentralised symbolic diagnosis approach. In *19th European Conference on Artificial Intelligence (ECAI-10)*, pages 99–104, Lisbon, Portugal, 8 2010.

[7] 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.

[8] X. Su and A. Grastien. Diagnosis of discrete event systems by independent windows. In *24th International Workshop on Principles of Diagnosis (DX-13)*, pages 148–153, 2013.

[9] X. Su and A. Grastien. Verifying the precision of diagnostic algorithms. In *21st European Conference on Artificial Intelligence*, 2014. in press.

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

[11] G. Lamperti and M. Zanella. On monotonic monitoring of discrete-event systems. In *18th International Workshop on Principles of Diagnosis (DX-07)*, pages 130–137, 2007.

[12] A. Schumann, Y. Pencolé, and S. Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In *22nd Conference on Artificial Intelligence (AAAI-07)*, pages 335–340, 2007.

[13] L. Brandán-Briones, A. Lazovik, and Ph. Dague. Optimal observability for diagnosability. In *19th International Workshop on Principles of Diagnosis (DX-08)*, pages 31–38, 2008.