Window-based Diagnostic Algorithms for Discrete Event Systems and Verifying Precision of Diagnostic Algorithms

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

Diagnosis of Discrete Event Systems (DES)
1. Diagnosis by computing belief states
2. Off-line computation: number of belief states makes it inapplicable for real-world problems
3. Symbolic and propositional logic using Binary Decision Diagram is subject to exponential blow-up in space.
4. Pre-computation of belief states takes exponential running time and has an exponential size in the number of states.

New Time-Window Algorithms
1. Time windows only consider most recent observations
2. Motivations and benefits:
   (1) flexibility: independent diagnosis analyses on separate time windows and skips irrelevant time windows
   (2) reduce diagnosis complexity: more manageable and build a diagnoser of polynomial size
   (3) precision loss? Precision is tested.
3. DES model: Figure 1 shows an Automaton
4. Diagnosis indicates whether system is in nominal mode or in faulty mode. Diagnoser assumes that system is not faulty unless proved otherwise.
5. Table 1 and 2 show two examples Window-based Diagnosis and demonstrate the importance to decide which algorithm to use and size of time window.

Verify precision of diagnosis algorithms using simulation
1. Measure precision of Time-Window Algorithms
2. Build simulation $s_i(M, A)$ for model $M$ and diagnostic algorithm $A$. Figure 2 illustrates $A_1$ simulation for DES model in Figure 1.

Experiments and results
1. Use Binary Decision Diagram to test diagnosability of model and precision of windows-based algorithms.
2. Example of factory operations: Figure 3 shows central model $M_c$ dispatching a job $(a_i)$ to operation plants $(M_i)$ and receiving feedback $(e_i)$. If $M_c$ enters faulty scenario, only $e_1$ will be observed from $M_1$.
3. In Figure 4, results show that they are all diagnosable.

Future work
1. “Backbone” diagnosis: remember what we know for sure
2. How to find root cause of ambiguity?
3. Create a benchmark for experiments.

Table 1: Algorithm 1 ($A_1$) slices a sequence of observations every 4 observations.

<table>
<thead>
<tr>
<th>Observation</th>
<th>Slice</th>
<th>Diagnosis</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a B C a</td>
<td>Y</td>
<td>Y</td>
<td>$F$</td>
</tr>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Algorithm 2 ($A_2$) slices a sequence of observations every 4 observations and time windows overlap. $A_1$ has drawbacks of imprecise diagnosis as it could not diagnose fault in the third observation.

<table>
<thead>
<tr>
<th>Observation</th>
<th>Slice</th>
<th>Diagnosis</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>a B C a b C</td>
<td>Y</td>
<td>$F$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. DES Model: $F$ is a faulty state. Other states are nominal. $a$, $b$, $c$ are observable events. $u$, $v$ are unobservable.

Figure 2. Part of Algorithm 1 simulation: Dotted lines also need to link $A_1$ to $A_0$, $B_0$, $C_0$ and $D_0$. Same applies to $B_1$ and $C_1$.

Figure 3. Example of factory operations

Figure 4. Running time of precision tests

From imagination to impact