Fault Diagnosis in Discrete-Event Systems: How to Analyse Algorithm Performance?

Yannick Pencolé [1]

Abstract. This paper addresses the problem of analyzing the performance of algorithms that solve the fault diagnosis problem in discrete-event systems in an experimental way. To achieve this purpose, we define a set of metrics for algorithm comparison on one hand. On the second hand, we propose an experimental platform based on a tool called DIADES (Diagnosis of Discrete-Event Systems) to run experiments and measure different algorithms.

1 INTRODUCTION

Fault Diagnosis in Discrete-Event Systems is the problem of determining the set of fault events that have occurred in a dynamic event-driven system usually based on a sequence of observed events. In the literature, there are many works and algorithms to solve this problem either from the AI community or the Automatic Control community. Most of these algorithms have been developed in order to solve special cases of system by introducing some assumptions or special properties (distributed systems [2, 12], coordinated systems [4], centralized diagnosis [15], decentralized diagnosis [12], distributed diagnosis [2], flexible diagnosis [11],...). The main drawback of these methods is not really in the methods themselves but is the lack of a generic method to experimentally compare these approaches to solve the same problems. The kind of question raised by this paper is: given one diagnosis problem, given the set of available algorithms that are able to solve this problem, which one is experimentally the best?

This is not the first time that this question has been asked: the DXC competition is definitely an attempt to answer the question [7]. However, in this paper, we adopt a complementary point of view. Instead of analyzing one real system as DXC proposes, we want to analyze algorithms on a specific class of systems (that is the class of discrete-event systems) independently from a model of any underlying and specific system. In the Model-Based diagnosis principle, there is a clear distinction between the Model and the algorithm that runs the Model [10]. Algorithm performance is affected by both the quality of the model and the algorithm itself, however the way to measure this performance is different. The quality of the model can be measured by correctness and diagnosability analyses whereas the intrinsic performance of an algorithm can be measured only by comparing it to the optimal algorithm on the *same* model whatever the quality of this model.

In order to find an answer to the previous question, it is first required to define what '*best*' means and a way to define why an algorithm would perform '*better*' than another one: this is the first contribution of this paper. We introduce a set of metrics for the specific problem of Fault Diagnosis in Discrete-Event Systems that can be exploited to compare the algorithmic performances. The second contribution of this paper is to propose a framework to define common problems (benchmarks) that can be freely utilized by the community. As opposed to many contribution on Model-Based diagnosis, we propose here to rely on randomly generated problems to measure the algorithm performance: as explained above, we are interested here in comparing the intrinsic performance of algorithms on the same model, whatever the quality of that model.

This paper is organized as follows. Section 2 first recalls the formal definition of the problem in the classical manner. Section 3 discusses the notion of performance of diagnosis algorithms that tend to solve the problem. Section 4 introduces some metrics whose objective is to measure the performance of algorithms. Section 5 describes some simple experiments, it firstly describes how the studied problem has been randomly generated by DIADES and secondly illustrates the proposed metrics by comparing the performance of two basic algorithms ('Forward breadth-first search' and 'Forward depth-first search').

2 FAULT DIAGNOSIS PROBLEM

This section formally recalls the fault diagnosis problem on dynamic discrete-event systems as defined for instance in [15].

2.1 Discrete-event system

A Discrete-Event System (DES) is a system whose behaviour is fully characterised at any time by a sequence of events that has occurred from an initial state. In this paper, the system is supposed to be a set of N components $\mathrm{Sys} = \{\mathrm{Comp}_1, \ldots, \mathrm{Comp}_N\}$ that are communicating with each other by synchronized events. There are many formalisms to represent such a system ([15], communicating automata [9], process algebra [3], Petri nets [2]. . .). Without loss of generality, the formalism chosen in this paper is synchronized automata.

Definition 1 (Model of a component) *The model of a component* Comp_i *is an automaton* $\mathcal{A}(\mathrm{Comp}_i) = (Q_i, \Sigma_i, T_i, q_{0i})$ *where*

- Q_i *is a finite set of states;*
- Σ_i *is the finite set of events;*
- $T_i \subseteq Q_i \times \Sigma_i \times T_i$ *is the finite set of transitions;*
- q_{0i} *is the initial state.*

A *trace* $\tau = e_1 \ldots e_k$ of a component Comp_i is a possible sequence of events such that there exists a finite transition path from state q_{0i} in the automaton $\mathcal{A}(\mathrm{Comp}_i)$: $q_{0i} \xrightarrow{e_1} q_1 \ldots q_{j-1} \xrightarrow{e_j} q_j \ldots q_{k-1} \xrightarrow{e_k} q_k$. Among the set of events Σ_i, two subsets are distinguished:

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France Univ de Toulouse, France, F-31400 Toulouse, France, email: yannick.pencole@laas.fr

1. Σ_i^{obs} is the set of observable events: an observable event, once it occurs, is supposed to be recorded by the diagnosis process;
2. Σ_i^{int} is the set of interacting events: if it occurs, an interacting event occurs at least in two components simultaneously and thus represents a communication between these two components.

No assumption is made about whether an interacting event is observable or not, however, for the sake of simplicity here as it is not the scope of the paper, we suppose that any interacting event e between Comp_i and Comp_j ($e \in \Sigma_i^{int} \cap \Sigma_j^{int}$) that is observable on Comp_i ($e \in \Sigma_i^{obs}$) is also observable on Comp_j ($e \in \Sigma_j^{obs}$). Any interacting event e is associated to a set of components that simultaneously interact each other so any interacting event implicilty represents a structural *connection* between these components. A connection between a set of components $\{\mathrm{Comp}_{j_1}, \ldots, \mathrm{Comp}_{j_l}\}$ exists iff there exists an interacting event e that belongs to every component of this set and only to these ones. The set of connections between the components is called the *structural model* of the system, also called the *topology* . The model of the system results from the synchronised product of the component model.

Definition 2 (Model of a system) *The model of the system is the automaton* $M = \mathcal{A}(\mathrm{Sys}) = (Q, \Sigma, T, q_0)$ *where*

- $Q \subseteq Q_1 \times Q_N$ *is the finite set of global states;*
- $\Sigma = \bigcup_{i=1}^{N} \Sigma_i$ *is the finite set of events;*
- $T \subseteq Q \times \Sigma \times Q$ *is the finite set of global transitions;*
- $q_0 = (q_{01}, \ldots, q_{0N})$ *is the global initial state.*

The $(q_1, \ldots, q_N) \xrightarrow{e} (q'_1, \ldots, q'_N)$ belongs to T iff

1. e is not an interacting event and there exists one and only one $i \in \{1, \ldots, N\}$ such that $q_i \xrightarrow{e} q'_i \in T_i$ and for any $j \in \{1, \ldots, N\} \setminus \{i\}$ $q_j = q'_j$;
2. e is an interacting event and belongs to the components $\{\mathrm{Comp}_{i_1}, \ldots, \mathrm{Comp}_{i_k}\}$ and for any $j \in \{i_1, \ldots, i_k\}$ $q_j \xrightarrow{e} q'_j \in T_j$ and for any $j \in \{1, \ldots, N\} \setminus \{i_1, \ldots, i_k\}$,$q_j = q'_j$.

Finally, in the following Q and T will respectively denote the set of global states and transitions that are reachable from the initial state q_0. Straightforwardly, Σ^{obs} denotes the set of observable events $\bigcup_{i=1}^{N} \Sigma_i^{obs}$ and a *global trace* $\tau = e_1 \ldots e_k$ of the system Sys is a possible sequence of events such that there exists a finite transition path from state q_0 in M: $q_0 \xrightarrow{e_1} q_1 \ldots q_{j-1} \xrightarrow{e_j} q_j \ldots q_{k-1} \xrightarrow{e_k} q_k$.

2.2 Diagnosis problem and solution

Now we are ready to define the classical Fault diagnosis problem on DES.

Definition 3 (Fault) *A* fault *is a non-observable event* $f \in \Sigma$.

A fault is represented as a special type of non-observable event that can occur on the underlying system. Once the event has occurred, we say that the fault is *active* in the system, otherwise it is *inactive*.

Definition 4 (Diagnosis problem) *A diagnosis problem is a triple* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *where* SD *is the global model of a system* Sys*,* OBS *is the set of timed observations* $\{(o_1, t_1), \ldots, (o_m, t_m)\}$ *with* $t_i \in \mathbb{R}^+, t_i < t_{i+1}$, $o_i \in \Sigma^{obs}$*, and* FAULTS *is the set of fault events defined over* SD.

Informally speaking, $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ represents the problem of finding the set of active faults from FAULTS that has occurred relying on the model SD and the sequence of observations OBS.

Definition 5 (Diagnosis Candidate) *A diagnosis candidate is a couple* (q, F) *where* q *is a global state of* M *(*$q \in Q$*) and* F *is a set of faults.*

A diagnosis candidate represents the fact that the underlying system is in state q and the set F of faults has occurred before reaching state q.

Definition 6 (Solution Diagnosis) *The* solution Δ *of the problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *is the set of diagnosis candidates* (q, F) *such that there exists for each of them at least one global trace* τ *of* SD *such that:*

1. *the observable projection of* τ *is exactly the sequence* $o_1 \ldots o_m$ *and the last event of* τ *is* o_m*;*
2. *the set of fault events that has occurred in* τ *is exactly* F*;*
3. *the final state of* τ *is* q.

Informally, candidate (q, F) is part of the solution if it is possible to find out in SD a behaviour of the system satisfying OBS which leads to the state q after the last observation of OBS and in which the faults F have occurred.

3 Diagnosis Algorithms

There exist many algorithms to solve the fault diagnosis problem presented above [15], [9], [18], [12], [16], [5], [6]. The purpose of some of them is to perform *off-line diagnosis* (also called *post-mortem diagnosis*): they consider OBS as a fixed set of observations when the algorithm starts and their objective is to find the solution of the problem. On the other side, some algorithms are so-called *on-line* in the sense that when the algorithm starts, the underlying system is operating, at a given time OBS is partly known and the purpose of the algorithm is not only to provide in the end the solution to the problem $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ but also to provide through the operating time of the system some *diagnosis updates*. Depending on the type of algorithms used, some performance indicators defined below may be pertinent or not to measure the performance of an algorithm. Performance of on-line algorithms especially requires a set of performance indicators related to time and diagnosis update capabilities that are not pertinent in the case of off-line diagnosis algorithms. Before introducing the performance indicators, a set of definitions related to the result provided by any algorithms is required.

Definition 7 (Current Diagnosis) *The* current diagnosis *of an operating algorithm* ALG *at time* t *is the set of candidates that* ALG *provides at time* t *if it is available, or the most recent set of candidates provided by* ALG *before* t.

Usually, off-line algorithms provide two set of candidates. The first set is provided when the algorithm starts and usually consist of $(q_0, \varnothing)$ whereas the second set is provided when the algorithm ends. However, some off-line algorithms may be incremental and thus provide more updates. On-line algorithms, as opposed to off-line algorithms, must provide updated sets of candidates so the current diagnosis is changing over the time.

Definition 8 (Final Diagnosis) *The* final diagnosis *of an algorithm* ALG *is the* current diagnosis *when the algorithm ends.*

The final diagnosis of an algorithm is the final result and thus should provide the solution of the problem.

3.1 Off-line algorithm performance

Since the underlying system Sys is real and has really emitted the set of observations OBS, any diagnosis problem is associated to the real trace τ_{real} of the system and, as a consequence the real diagnosis of the system.

Definition 9 (Real Diagnosis) *The real diagnosis is the candidate* (q_{real}, F_{real}) *associated to the real trace* τ_{real}.

Determining (q_{real}, F_{real}) or at least F_{real} is the main objective of any diagnosis algorithms but to reach this objective, there are many obstacles. First, Definition 4 defines a model-based diagnosis problem that relies on the existence of a sound model SD so that τ_{real} is assumed to be represented in the model SD. However, in practice as described in [1], this may be not the case so the algorithm may not be able to determine τ_{real} as a possible trace which means that (q_{real}, F_{real}) may not be representable in SD at all.

Definition 10 (Sound problem) *The problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *is sound if* τ_{real} *is represented in the problem.*

In other words, if the problem is sound, the solution diagnosis contains the real diagnosis.

Definition 11 (Sound Algorithm) *The algorithm* ALG *is sound with respect to the problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *if the solution diagnosis is contained in the final diagnosis.*

A sound algorithm ALG always provides a superset of the solution diagnosis which guarantees that if the problem is sound, ALG provides the real diagnosis.

Definition 12 (Accurate Algorithm) *The algorithm* ALG *is accurate with respect to the problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *if the final diagnosis is exactly the solution diagnosis.*

An accurate algorithm is a particular sound algorithm that returns the smallest superset of the solution diagnosis. An accurate algorithm takes benefit of all the observations to refine the candidates and provide the smallest but still correct refinement.

Second, even if the real trace is represented in SD, the lack of observable events may lead to many ambiguities, which means that the diagnosability level of the algorithm is poor.

Definition 13 (Diagnosable problem) *The problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *is diagnosable if the solution diagnosis contains only one candidate.*

Definition 14 (Fault-Diagnosable problem) *The problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *is fault-diagnosable if the solution diagnosis contains only candidates as* $\{(q_1, F), \ldots, (q_k, F)\}$.

In other words, if the problem is fault-diagnosable then the occurrence of the faults F are certain, only the state estimate is ambiguous.

Property 1 *If the problem is sound and diagnosable, the final diagnosis of any accurate algorithm on this problem is the real diagnosis.*

This property provides the skeleton for the design of performance indicators. An algorithm that optimally performs on a given problem $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ requires soundness and diagnosability analyses on one hand as proposed for instance in [15], [14], [13], [1] and accuracy analyses on the other hand, which is the main concern of this paper.

3.2 On-line algorithm performance

An on-line algorithm must perform as well as an off-line when it ends, which means that the above definitions and properties also hold for on-line algorithms. However, the second challenge of on-line algorithms is to propose diagnosis updates through time relying on a flow of observations that leads to the full sequence of OBS. In other words, on-line algorithms should solve a finite-set of diagnosis problems derived from the initial problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$. At time t_i, a new observation o_i is received and the on-line algorithm should then provide a diagnosis update at time t_i that is the solution of the problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_i)$ where $\mathrm{OBS}_i = \{(o_j, t_j) \in \mathrm{OBS} \wedge t_j \leq t_i\}$.

Definition 15 (Perfect On-Line Algorithm) *An on-line algorithm* ALG *is* perfect *iff* $\forall i \in \{1, \ldots, |\mathrm{OBS}|\}$*, the current diagnosis of* ALG *at time* t_i *is the solution of* $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_i)$.

Such an algorithm is called *perfect* as there is no other algorithms that can perform better: at any time, it provides the solution for the available set of observations. Such an algorithm exists and is defined in [Sampath95] and is called the diagnoser. The diagnoser for a given system description SD and a given set of faults FAULTS is a deterministic finite-state machine obtained by a complete knowledge precompilation: each state of the diagnoser provides the current diagnosis and each transition is labeled with an observable event of the system. To solve the diagnosis problem with the diagnoser, the underlying algorithm simply builds the unique transition path from the initial state of the diagnoser (the initial diagnosis) associated to the given sequence of observations OBS, the state ending this path provides the final diagnosis. For the observation (o_i, t_i) is the sequence OBS, it is sufficient to trigger one transition to get a diagnosis update, which is in constant time and *almost* instantaneous. Moreover, by the construction of the diagnoser during the precompilation stage, any diagnoser state reached by a transition path from the initial state associated to the sequence of observations OBS_i is the solution of the problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_i)$.

Now, why not keeping using this diagnoser to solve any online diagnosis problem if it is perfect? The reason is that the precompilation may not be practically feasible: recalling that N is the number of components in the system, the size of the machine to build is in the worst case in $o(2^{2^N})$ and in $o(2^{|\mathrm{FAULTS}|})$.

4 PERFORMANCE METRICS

This section lists a set of performance metrics. These indicators are generic in the sense that most of them compare any type of algorithm results two by two. In particular, it does not necessarily rely on the real diagnosis that is usually unknown unless further analyses on the system are perform or the faulty scenario results from a simulation in which faults have been injected [7]. If the real diagnosis Δ_{real} is

known then it can be utilized as any other diagnosis to measure the performance of the available algorithms. As detailed in this section, the proposed metrics, once they are applied on the real diagnosis Δ_{real}, are equivalent to some metrics also proposed in [7].

4.1 Time-Independent generic metrics

Time-Independent metrics take into account two diagnoses (a set of diagnosis candidates) Δ_1 and Δ_2 resulting from the diagnosis problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$ and provide some ways to compare them. Let $\Delta_1 = \{(q_{i_1}, F_{i_1}), \ldots, (q_{i_l}, F_{i_l})\}$ and $\Delta_2 = \{(q_{j_1}, F_{j_1}), \ldots, (q_{j_k}, F_{j_k})\}$ be two diagnoses on a given diagnosis problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$.

Definition 16 (Belief State Distance) *The Belief State Distance* $\mathrm{BSD}(\Delta_1, \Delta_2)$ *is:*

$$|\{q, \exists (q, F) \in \Delta_1 \setminus \Delta_2\}| + |\{q, \exists (q, F) \in \Delta_2 \setminus \Delta_1\}|.$$

The metrics BSD relies on the so-called Hamming distance: it counts the number of differences between the two belief states $\{q_{i_1}, \ldots, q_{i_l}\}$ and $\{q_{j_1}, \ldots, q_{j_k}\}$ so it behaves as a distance that is null iff $\{q_{i_1}, \ldots, q_{i_l}\} = \{q_{j_1}, \ldots, q_{j_k}\}$. The same type of metrics can be define to compare the fault candidates only.

Definition 17 (Fault Candidate Distance) *The Fault Candidate Distance* $\mathrm{FCD}(\Delta_1, \Delta_2)$ *is*

$$|\{F, \exists (q, F) \in \Delta_1 \setminus \Delta_2\}| + |\{F, \exists (q, F) \in \Delta_2 \setminus \Delta_1\}|.$$

When performing off-line diagnosis, we may not be interested in estimating the current belief state as it is not necessary to provide further diagnosis updates, in this case then, BSD is not the most important metrics to look at, but it is FCD as it is the one that only takes into account the difference between fault candidates.

These previous two metrics compare the diagnoses as a difference of information and has the advantage to be a mathematical distance in order to measure how far from each other the diagnoses are. However, this distance does not compare carefully the common candidates. Hence the introduction of the notion of accuracy and precision.

Definition 18 (Accuracy)

$$\mathrm{Acc}(\Delta_1, \Delta_2) = \frac{|\Delta_1 \cap \Delta_2|}{|\Delta_1 \cup \Delta_2|}$$

$\mathrm{Acc}(\Delta_1, \Delta_2)$ is a ratio that states the number of common candidates in Δ_1 and Δ_2. It tends to 1 when $\Delta_1 = \Delta_2$, it tends to 0 if no common candidate exists. This metrics can especially be utilized when the diagnosis Δ_{real} is known. Indeed, for any diagnosis Δ_1, $\mathrm{Acc}(\Delta_1, \Delta_{real})$ is 0 if Δ_1 does not contain the real candidate. It can also be utilized when the theoretical solution (Definition 6) of $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$ is known: any accurate algorithm providing the diagnosis Δ' is necessarily such that $\mathrm{Acc}(\Delta, \Delta') = 1$. Last but not least, by analyzing $\mathrm{Acc}(\Delta, \Delta_{real})$, if it is zero, it also means that the model is not sound and needs to be redesigned [1].

Definition 19 (Precision)

$$\mathrm{Prc}(\Delta_1, \Delta_2) = \min\left(\frac{|\Delta_1|}{|\Delta_2|}, \frac{|\Delta_2|}{|\Delta_1|}\right).$$

The precision checks the ratio between the number of candidates. If $\mathrm{Prc}(\Delta_1, \Delta_2) = 1$, it means that Δ_1 and Δ_2 have the same precision. If the precision is lower than 1, the closer it is to 0, the higher the difference of ambiguity is between Δ_1 and Δ_2. By comparing a diagnosis Δ_1 to the solution Δ of $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$, we can measure the precision level of the algorithm that provided Δ_1. Two cases hold:

1. $|\Delta_1| > |\Delta|$ means that the algorithm is not as precise as the accurate algorithm. In this case, it means that the algorithm does not take benefit of the full observability represented in SD. It may be due to the fact the algorithm is computing an approximation of the solution.
2. $|\Delta_1| < |\Delta|$ means that the algorithm only returns partial solutions in the sense that even if the problem is sound, the solution provided by the algorithm may be just wrong.

Precision and accuracy are complementary metrics, as precision does not measure anything about common diagnosis. A less accurate diagnosis may be more precise than a more accurate diagnosis. Ultimately, for a given sound problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$ and its solution Δ, the objective is to design an algorithm that provides Δ' such that $\mathrm{BSD}(\Delta', \Delta) \to 0$, $\mathrm{FCD}(\Delta', \Delta) \to 0$, $\mathrm{Acc}(\Delta', \Delta) \to 1$ and $\mathrm{Prc}(\Delta', \Delta) \to 1$.

4.2 Time metrics

Definition 20 (Horizon) *The Horizon* $H_{\mathrm{ALG}}(i)$ *is the time duration required by the on-line algorithm* ALG *to solve* $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_i)$.

The Horizon of a perfect diagnosis algorithm is $\forall i, H(i) = 0$. The Horizon measures the capability of an on-line algorithm to *follow* the flow of observations. For example, if $H_{\mathrm{ALG}}(i) > t_{i+1}$, then ALG is not able to provide a final diagnosis for the problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_i)$ before it has to solve $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS}_{i+1})$. The consequence is that ALG can have cumulative delays which show that ALG has weak on-line performances.

4.3 Reality metrics

In [7], other metrics are defined. As opposed to the one presented above, they require the knowledge of the real diagnosis. Such metrics measure the quality of the couple $(\mathrm{SD}, \mathrm{ALG})$ as a whole, in the sense that to improve the values measured by these metrics, either SD or ALG or both must be improved. Especially, the improvement of SD usually consists of improving the soundness of the model [1] or the diagnosability of the model and the underlying system by sensor placement [17],[13].

Definition 21 (False Positive) *The number of false positive (*FP*) is the number of faults in* Δ_{ALG} *that are not in* $\Delta_{real} = (q_r, F_r)$*:*

$$\mathrm{FP} = |\{f \in \mathrm{FAULTS}, \exists (q, F) \in \Delta_{\mathrm{ALG}} \wedge f \notin F_r\}|$$

FP usually indicates a lack of diagnosability.

Definition 22 (False Negative) *The number of false negative (*FN*) is the number of faults in* Δ *such that:*

$$\mathrm{FN} = |\{f \in F_r, \forall (q, F) \in \Delta_{\mathrm{ALG}}, f \notin F\}|.$$

FN usually indicates a lack of soundness.

5 FIRST EXPERIMENTS

In this section, we present a set of experiments about two basic algorithms denoted FBFS (Forward Breadth-First Search) and FDFS (Forward Depth-First Search) in order to show the effectiveness of the presented metrics. As stated above, we are not concerned in this section about the quality of the model SD, OBS such as diagnosability and soundness that are out of scope of this paper and requires simulated scenarios and a fault injection simulator to compute for instance the accuracy of the algorithm with respect to the real diagnosis. In these experiments, SD is assumed to be sound (the real trace is represented in the model). The purpose is to compare some algorithms with each other on the same diagnosis problem and to compare their performance.

5.1 Algorithm descriptions

The algorithms to run these first experiments have been chosen for their simple description and not for their efficiency. The aim here is to show how the metrics are able to characterize the behaviour of algorithms. As explained in Section 2.2, the starting point of both algorithms is N automata and their synchronization law which implicitly represent SD, a given set of faults FAULTS to diagnose and the sequence of observations $\text{OBS} = \{(o_1, t_1), \ldots, (o_m, t_m)\}$.

5.1.1 Forward breadth-first search: FBFS

The principle of the FBFS algorithm is to start from an initial belief state $\mathcal{I}$[2] and searches with help of a Breadth-First Search (BFS) strategy the set of traces of the global model M that can emit OBS and extracts from them the diagnosis candidates. Note that by construction this algorithm is sound as the BFS guarantees it will find out the complete set of traces. The search space (similar to the behavioural space in [9]) is recursively defined as follows. For each candidate of $(q, F) \in \mathcal{I}$, we associate the state $s = (q, F, 0)$ where 0 means that no observation has occurred. Then, let $s = (q, F, k)$ be a state of the search space then $s' = (q', F', k')$ is a state of the search space iff there exists a transition $q \xrightarrow{e} q'$ in M and one of the following condition holds:

1. e is observable, $F = F'$ and $k' = k + 1$;
2. $e \in$ FAULTS, $F' = F \cup \{e\}$ and $k' = k$;
3. e is neither faulty nor observable and $F = F$ and $k' = k$.

The set of states s' defined as above is denoted $next(s, e)$. Finally FBFS is simply defined as follows by the use of a queue (First In First Out):

```
FBFS(M,FAULTS,OBS,I): output D
do
  for(s=(q,F,0) : (q,F) in I)
    push(queue,s);
    visited(s) <- true;
  endfor

  while(not empty(queue))
    s=(q,F,k) <- pop(queue);
    if(k=m) then insert((q,F),D);
    else
      for(s' : next(s,e), e event of M)
        if ((e = o_k or not_observable(e))
            and not (visited(s')))
        then
          push(queue,s'); visited(s') <- true
        endif
      endfor
    endif
  endwhile
done
```

[2] In Section 2.2, we supposed the initial belief state is $\{(q_0, \emptyset)\}$ however, here, for the sake of generality, we do not make any assumption about it, it can be any set of diagnosis candidates.

5.1.2 Forward depth-first search: FDFS

The principle of the FDFS algorithm is similar to the one of FBFS except that the search strategy relies on a Depth-First Search. As for its counterpart, this algorithm is sound. The main difference is the use of a stack (Last In First Out) instead of a queue.

```
FDFS(M,FAULTS,OBS,I): output D
do
  for(s=(q,F,0) : (q,F) in I)
    push(stack,s);
    visited(s) <- true;
  endfor

  while(not empty(stack))
    s=(q,F,k) <- pop(stack);
    if(k=m) then insert((q,F),D);
    else
      for(s' : next(s,e), e event of M)
        if ((e = o_k or not_observable(e))
            and not (visited(s')))
        then
          push(stack,s'); visited(s') <- true
        endif
      endfor
    endif
  endwhile
done
```

5.2 Generation of the Diagnosis Problem

In this paper, as opposed to usual Model-Based Diagnosis papers, we decide to run experiments on a purely random Diagnosis Problem. In order to generate this problem, we develop a random generator for Diagnosis Problem with help of the DIADES tools developed in CNRS-LAAS [8]. The generation of the illustrating model is in three stages.

1. Topology generation: the topology (see Section 2.1) is randomly chosen with help of the tool `generate_topology`. The topology of this example has been generated by fixing the number of components to 10.
2. Component generation: for each component of the topology, the behaviour of this component (see Definition 1) is randomly generated with help of the tool `des_generate` that takes as input the previously generated topology and thus generates a set of 10 automata (see Table 1) and their synchronization model (see Section 2.1). Most importantly, `des_generate` generates automata that are globally consistent in the sense that the global model exists and each local transition of each automaton belongs at least once to the global model. The size of the global model is big enough so that

Diades (with its tool `global_model`) was not able to compute it due to lack of computational resources (memory resources) [3], it has given up after building 185 000 states and 1 000 000 transitions (theoretically, the number of states for this example is lower than 10^{10} which the number of states in the free product of the 10 automata).

3. To finally define FAULTS and OBS, we use the tool `simulate` that takes as input the set of 10 automata and randomly generates a scenario of a fixed observable size. The scenario utilized in the following example consists of a sequence of 10 observations. To define FAULTS, we selected a set of 9 non-observable events as faults.

Comp	StateNb	TransNb	Comp	StateNb	TransNb
$C0$	7	16	$C1$	20	46
$C2$	5	15	$C3$	20	45
$C4$	9	22	$C5$	8	12
$C6$	15	31	$C7$	5	17
$C8$	18	42	$C9$	8	23

Table 1. Size of the randomly generated automata for this example

5.3 First Results

This subsection presents some experiments that illustrate performance metrics introduced in Section 4.1 and the way to use them for comparing the behaviour of diagnosis algorithms to each other. Figures 1, 2, 3, 4 present the behaviour of the metrics of Section 4.1 with the algorithms FBFS and FDFS for the diagnosis problem $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$ that is described in the previous section. For every Figure, there are three curves. For a given metrics m, the thick curve represents $m(\Delta_{\mathrm{FDFS}}(t), \Delta)$ where Δ is the solution of $(\mathrm{SD}, \mathrm{FAULTS}, \mathrm{OBS})$ (see Definition 6) and $\Delta_{\mathrm{FDFS}}(t)$ is the partial diagnosis (a set of candidates) that FBFS is able to provide at time t. The thin plain curve represents $m(\Delta_{\mathrm{FBFS}}(t), \Delta)$ and finally the dashed curve represents $m(\Delta_{\mathrm{FDFS}}(t), \Delta_{\mathrm{FBFS}}(t))$.

Figure 1 illustrates the Belief State Distance. At time 0, the distance between the initial diagnosis $(q_0, \varnothing)$ and the solution is in this case 230. At time 50, FDFS starts to produce new candidates that improves the BSD as it is getting closer to the solution. The FDFS curve really shows the overall behaviour of a DFS strategy, when the BSD is stable for a while, it means that FDFS keeps backtracking. As opposed to FDFS, FBFS takes longer to provide the first candidates and the final solution.

This difference can be clearly seen on $\mathrm{BSD}(\Delta_{\mathrm{FDFS}}(t), \Delta_{\mathrm{FBFS}}(t))$ that keeps growing from time 40 to time 80 which shows that FBFS and FDFS are drastically different algorithms. FBFS accelerates the computation of candidates in the end whereas DFS provides candidates with a more stable speed.

Figure 2 illustrates the Fault Candidate Distance. As opposed to BSD,FCD is very small at initial time. It is due to the fact that the initial diagnosis $(q_0, \varnothing)$ has an empty set of fault candidates whereas the solution diagnosis contains 9 fault candidates only (but many possible states associated to them). FDFS provides the correct fault candidates at time 30, whereas FBFS provided them at time 50.

[3] At present, the way the tool `global_model` is implemented is far from being optimal.

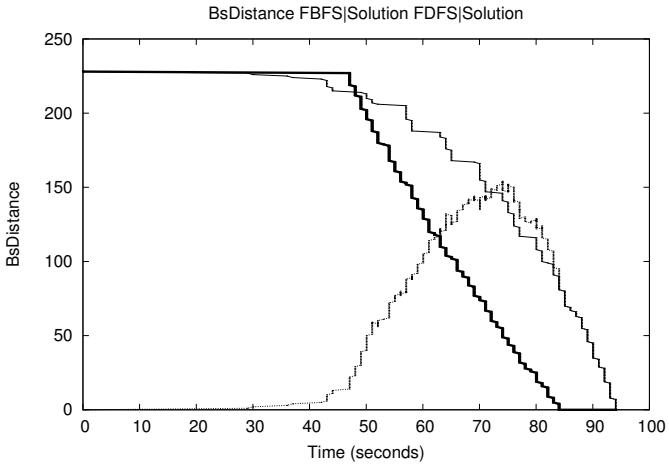

Figure 1. BSD metrics

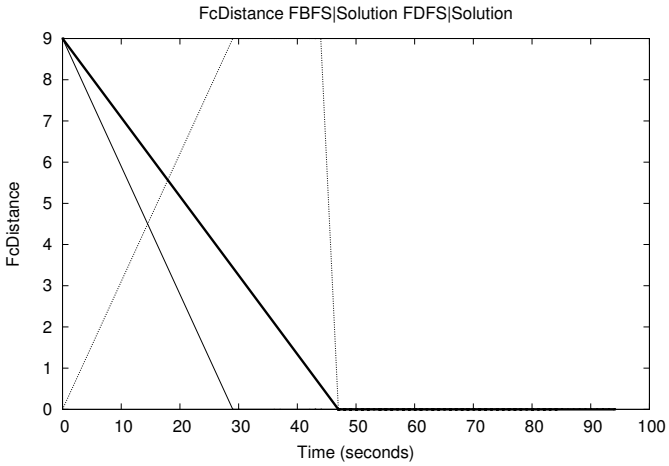

Figure 2. FCD metrics

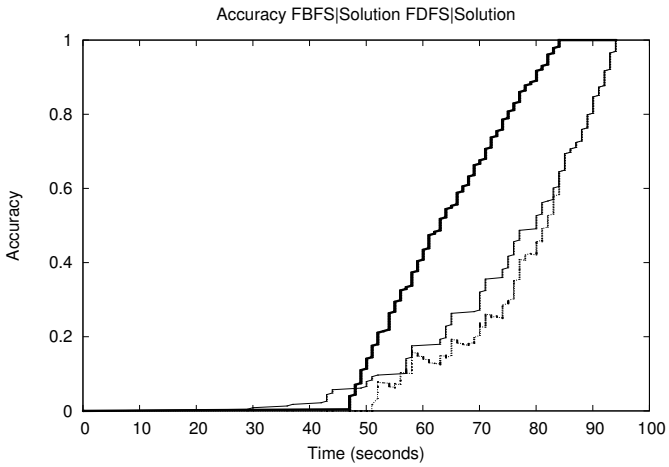

Figure 3. Accuracy metrics

Figures 3 and 4 respectively describe the overall accuracy metrics and precision metrics (embedding belief states and fault candidates). As noticed by analyzing BSD and FCD, the accuracy of FDFS clearly improves earlier than the accuracy of FBFS. Both accuracy tend to 1 which results from the fact that both algorithms are sound on the given problem. Finally, $\mathrm{Acc}(\Delta_{\mathrm{FDFS}}(t), \Delta_{\mathrm{FBFS}}(t))$ also tends to one that shows both algorithms converge on the same solution. As far as the precision is concerned, the behaviour is rather similar to accuracy where FBFS and FDFS are compared to the solution Δ. The most interesting metrics is in this case $\mathrm{Prc}(\Delta_{\mathrm{FDFS}}(t), \Delta_{\mathrm{FBFS}}(t))$ which measures the relative precision of both algorithms over time. Due to the fact that the strategies of FBFS and FDFS are drastically different the behaviour of the relative precision is rather erratic till it converges to 1.

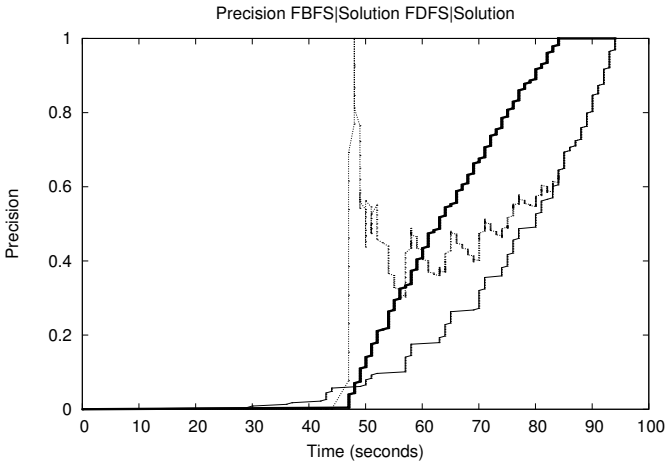

Figure 4. Precision metrics

6 CONCLUSION

In this paper, we discuss the problem of evaluating the performance of diagnosis algorithms that solve the Fault-Diagnosis problem from an experimental point of view. For this special problem, there is a clear distinction between the model part (a set of automata or any equivalent representation) and the algorithm that solves the problem. That is the main reason of introducing a set of proper metrics for algorithms only and for introducing a framework to randomly generate models as it is proposed with the tool DIADES. Of course the experiments presented here are very preliminary and we expect in the very close future to generate a full set of benchmarks and to analyze more sophisticated algorithms with the ones developed in our previous work [12],[16] as a priority. The model generator we are using defines a set of parameters (number of nodes, connectivity, number of states, number of observable events, . . .) that should be interesting to play with in order to generate very different type of DES models and thus sees how algorithms perform on them. There will be two main objectives, that, we actually believe, are essential for the MBD community:

1. to have a clear understanding about how the available diagnosis algorithms scale on the same problems;
2. and more importantly, to have a better understanding about how to improve the efficiency of the algorithms by closer comparative analyses of one another.

REFERENCES

[1] N. Belard, Y. Pencolé, and M. Combacau, 'A theory of meta-diagnosis: Reasoning about diagnostic systems', in *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11)*, pp. 731–737, (2011).

[2] A. Benveniste, E. Fabre, S. Haar, and C. Jard, 'Diagnosis of asynchronous discrete event systems, a net unfolding approach', *IEEE Transactions on Automatic Control*, **48**(5), 714–727, (2003).

[3] L. Console, C. Picardi, and M. Ribaudo, 'Process algebras for systems diagnosis', *Artificial Intelligence*, **142**(1), 19–51, (2002).

[4] R. Debouk, S. Lafortune, and D. Teneketzis, 'A coordinated decentralized protocol for failure diagnosis of discrete event systems', *Journal of Discrete Event Dynamical Systems: Theory and Application*, **10**, 33–86, (2000).

[5] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva, 'Diagnosis of discrete-event systems using satisfiability algorithms', in *American National Conference on Artificial Intelligence (AAAI-07)*, (2007).

[6] Pr. Kan John and Al. Grastien, 'Local consistency and junction tree for diagnosis of discrete-event systems', in *Eighteenth European Conference on Artificial Intelligence (ECAI-08)*, (2008).

[7] T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman, 'Towards a framework for evaluating and comparing diagnosis algorithms', in *20th International Workshop on Principles of Diagnosis (DX-09)*, pp. 373–381, (2009).

[8] Universit de Toulouse LAAS-CNRS. Diades: Diagnosis of discrete-event systems. http://homepages.laas.fr/ypencole/diades.

[9] G. Lamperti and M. Zanella, *Diagnosis of active systems*, Kluwer Academic Publishers, 2003.

[10] G. Lamperti and M. Zanella, 'Diagnosis of discrete-event systems by separation of concerns, knowledge compilation, and reuse', in *16th European Conference on Artificial Intelligence (ECAI-04)*, pp. 838–842, (2004).

[11] G. Lamperti and M. Zanella, 'Flexible diagnosis of discrete-event systems by similarity-based reasoning techniques', *Artificial Intelligence*, **170**(3), 232–297, (2006).

[12] 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*, **164**, 121–170, (May 2005).

[13] P. Ribot, Y. Pencolé, and M. Combacau, 'Design requirements for the diagnosability of distributed discrete event systems', in *19th International Workshop on Principles of Diagnosis (DX'08)*, Blue Mountains, Australia, (September 2008).

[14] J. Rintanen and A. Grastien, 'Diagnosability testing with satisfiability algorithms', in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 532–537, (2007).

[15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, 'Diagnosability of discrete event system', *IEEE Trans. on Automatic Control*, **40**(9), 1555–1575, (1995).

[16] A. Schumann, Y. Pencolé, and S. Thiébaux, 'A spectrum of symbolic on-line diagnosis approaches', in *22nd American National Conference on Artificial Intelligence (AAAI)*, pp. 335–340, (2007).

[17] L. Travé-Massuyès, T. Escobet, and R. Milne, 'Model-based diagnosability and sensor placement application to a frame 6 gas turbine subsystem', in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI'01*, volume 1, pp. 551–556, (2001).

[18] F. Xue, L. Yan, and D. Zheng, 'Fault diagnosis of distributed discrete event systems using obdd', *Informatica*, **16(3)**, 431–448, (2005).