Exhaustive Diagnosis of Discrete Event Systems through Exploration of the Hypothesis Space

Alban Grastien [1,2], **Patrik Haslum** [2,1], **and Sylvie Thiébaux** [2,1]

[1] *NICTA, Canberra Research Laboratory, Australia*
firstname.lastname@nicta.com.au
[2] *The Australian National University, Canberra, Australia*

Abstract

We formulate DES diagnosis as the problem of finding preferred candidates in a space of hypotheses. This formulation allows us to compute an exhaustive diagnosis by solving a series of simple problems, called "diagnosis tests", which consist in finding a candidate in a set of hypotheses. In this way, we can avoid enumerating all explanations of the observation. Experiments on alarm data generated by power networks show that this approach is promising to monitor large systems.

1 INTRODUCTION

Discrete-event systems (DES) are a popular model for dynamic systems when the states and the events can be represented with acceptable accuracy at a discrete level (Cassandras and Lafortune, 1999). The DES model usually includes *observable events* which correspond to the actual observables of the diagnosis problem. The purpose of diagnosis is to determine if the evolution of the system is nominal, and if not, what type of faulty behaviour it exhibits. In most realistic situations, the classification of behaviours (into nominal and different fault types) is proper, i.e., the purpose of diagnosis is not to reconstruct in complete detail all possible behaviours that may have taken place. Furthermore, there can be a preference relation over diagnoses (representing, e.g., likelihood), and the task of diagnosis is then to identify only the preferred explanations.

In spite of this, DES diagnosis problems have traditionally been solved by reconstructing all system traces consistent with the observation, either explicitly (Zanella and Lamperti, 2003; Aghasaryan *et al.*, 1998) or through a precompiled structure (Sampath *et al.*, 1995). This makes traditional DES diagnosis methods very sensitive to the size of the system. The combinatorial complexity of computing and representing all system traces that can explain the observation puts a tight limit on the class of systems that can be diagnosed. In order to monitor large networks, like power grids consisting of thousands of components, a different approach is required.

In contrast, diagnosis of circuit or continuous systems is typically performed by means of (simple) tests.[1] For instance, in (Reiter, 1987) (for circuits) and in (Dague, 1994) (for continuous systems), a *diagnosis hypothesis* is tested and either it is proved to be a *diagnosis candidate*, or a *conflict* is derived from the model and the observation and is used to generate more diagnosis hypotheses. Moreover, the most popular approach for continuous systems is to identify *possible conflicts* off-line (Pulido and Alonso González, 2002). A possible conflict is a Boolean test on the system observation such that the test returns *true* iff the corresponding conflict can be derived from the observation. The possible conflicts are tested on-line and the diagnosis is derived from the test outputs. In both cases, enumeration of all system behaviours consistent with the observation – even implicitly – is avoided. As far as we know, no such approach has been proposed for DES diagnosis, although similarities between DES and continuous systems have been identified (Cordier *et al.*, 2006).

In this paper, we propose an approach for solving DES diagnosis that is not based on following all system traces. Instead, we define simple Boolean questions called *diagnosis tests*, which, given a set of diagnosis hypotheses, indicate whether one of them explains the observation. Answering such a test requires only to find *one* trace. We show how DES diagnosis can be computed as a collection of tests and demonstrate its application on data generated by an electricity utility.

The paper is organized as follow. After background definitions (Section 2), we formulate DES

[1] Approaches to diagnosis of circuits have also been proposed that compute all possible states of the system (Darwiche, 1998).

diagnosis as a problem of finding particular elements in a hypothesis space (Section 3). In Section 4, we define the diagnosis test, and in Section 5 we present two possible families of strategies based on tests to explore the hypothesis space. In Section 6, we show how tests can be implemented using heuristic search or propositional satisfiability, and we provide results of experiments in Section 7.

2 BACKGROUND

We consider dynamic systems that are modelled as discrete-event systems. A (partially observable) DES is a tuple $\langle \Sigma, \mathsf{L}, obs \rangle$ where Σ represents the finite set of events that can take place in the system, $\mathsf{L} \subseteq \Sigma^*$ models the possible evolutions of the system, and *obs* is a function that associates each trace u of events with an observation $obs(u)$. (The notation X^* represents the Kleene closure of set X.) Language L captures the possible sequences of events in the system, i.e., sequence (or trace) $u = [e_1, \ldots, e_k]$ is possible according to the model iff $u \in \mathsf{L}$. We assume the model is complete, i.e., any sequence of events that is indeed possible in the system is present in language L.

Trace u is partially observed by the diagnosing system. To simplify the notations, we assume the observation of trace u is deterministically defined by a function $obs(u)$. The partial observability is typically modelled by a subset $\Sigma_o \subseteq \Sigma$ of observable events, and $obs(u) \in {\Sigma_o}^*$ is defined as the projection of u on the set Σ_o, i.e., $obs(u)$ is a copy of u where all events that are not observable are deleted.

In this paper, we will assume that the possible faults of the system are modelled by a set $\Sigma_f \subseteq \Sigma$ of fault events. We make no assumptions about the number occurrences of a fault (i.e., faults may be permanent, transient or intermittent), or about their diagnosability.

3 DIAGNOSIS AND THE HYPOTHESIS SPACE

Diagnosis is usually defined as the problem of determining the type of the behaviour that the system is exhibiting. We call the set $\mathcal{H}$ of possible behaviour types the *hypothesis space*. What elements $\mathcal{H}$ contains depends on what level of detail we require of the diagnosis. For example, if we are only interested in fault detection, $\mathcal{H}$ may simply have two elements: "nominal" and "faulty". More common is to consider each set of fault events that may have occurred as a distinct hypothesis, but there are also other hypothesis spaces that can be of interest in DES diagnosis (several examples are discussed below). What matters, from our perspective, is only that each trace $u \in \Sigma^*$ is associated with a single hypothesis, denoted $\delta(u)$. We call a hypothesis a *candidate* if it is associated with at least one trace that is consistent with the observation of the system, i.e., if it may be the actual system behaviour type.

In most cases, not every candidate is equally plausible or interesting. For example, if hypotheses are sets of fault events that have taken place, a hypothesis that is a strict superset of another candidate is usually assumed not to have occurred (following the principle of Occam's razor). Thus, we assume that there is a *preference relation*, denoted by $\preceq$, on $\mathcal{H}$, with $h \preceq h'$ meaning that hypothesis h is preferred to h'. Formally, $\preceq$ is a non-strict (i.e., reflexive) partial order on $\mathcal{H}$.

Definition 1. *Given a partially observable DES* $\mathsf{D} = \langle \Sigma, \mathsf{L}, obs \rangle$*, an observation* o*, and a hypothesis space* $\mathcal{H}$*, a hypothesis* $h \in \mathcal{H}$ *is a* diagnosis candidate *iff there exists* $u \in \mathsf{L}$ *such that* $obs(u) = o$ *and* $h = \delta(u)$ *The* model-based diagnosis $\Delta(\mathsf{D}, o, \mathcal{H})$ *is the set of candidates.*

A candidate h *is* minimal *(or* preferred*) iff there is no other candidate preferable to* h*, i.e.,* $\forall h' \in \Delta,\ h' \preceq h \Rightarrow h' = h$*. The* preferred diagnosis*,* $\min_{\preceq}(\Delta(\mathsf{D}, o, \mathcal{H}))$*, is the set of minimal candidates.*

In the rest of the paper, we write only Δ instead of $\Delta(\langle \Sigma, \mathsf{L}, obs \rangle, o, \mathcal{H})$ when the system, observation and hypothesis space are clear from context.

The techniques traditionally used to exhaustively solve diagnosis problems are beyond the scope of this paper. Two main approaches are used:

- The set *Expl* of traces u that generate o is computed, and the diagnostic result is extracted by analysing this set of traces.
- The model is compiled away in a structure from which diagnosis can be extracted in linear time with respect to the observation.

Whichever method is used, the time and/or memory required to compute an exhaustive diagnosis rises exponentially with the number of components, which limits the size of systems that can be diagnosed by these methods.

For non-exhaustive diagnosis, i.e., when the goal is only to find a single candidate, approaches based on reduction to SAT or to AI planning have proved quite efficient (Grastien *et al.*, 2007; Sohrabi *et al.*, 2010), since these search for a unique trace. The present work uses similar techniques to compute exhaustively the preferred diagnosis. In Section 4, we define the test operation in order to explore the hypothesis space and to generate the preferred diagnosis without computing *Expl*.

3.1 Examples of Hypothesis Spaces

The hypothesis space most commonly used in DES diagnosis is the *set space*, where the hypothesis associated with a trace is the set of fault events that appear in it, i.e., $\delta(u) = \{f \in \Sigma_f \mid f \in u\}$. Formally, the space is $\mathcal{H}_{\text{set}} = 2^{\Sigma_f}$, and the preference relation on $\mathcal{H}_{\text{set}}$ is defined by $h \preceq_{\text{set}} h' \Leftrightarrow h \subseteq h'$.

When intermittent faults are considered, we may be interested in the exact number of occurrences of each fault, not only in if it has occurred or not. In the *multiset space*, $\mathcal{H}_{\text{ms}}$, each hypothesis h is a function $\Sigma_f \to \mathbf{N}$ mapping each fault $f \in \Sigma_f$ to its number of occurrences. A hypothesis is preferred to another if it maps each fault

type to a smaller number, i.e., $h \preceq_{\text{ms}} h' \Leftrightarrow \forall f \in \Sigma_f,\ h(f) \leq h'(f)$.

However, this is not the only possible preference relation on multiset hypotheses. Cordier et al. (1997) consider the case where there is an order on the fault events, such that $f \prec f'$ if the occurrence of f is infinitely more likely than the occurrence of f'. The preference relation over fault multisets is then defined by $h \preceq_{\text{ms-lex}} h' \Leftrightarrow \big(\forall f \in \Sigma_f,\ h(f) > h'(f) \Rightarrow \exists f' \in \Sigma_f : f' \prec f \ \wedge\ h(f') < h'(f')\big)$.

Finally, if we want to discriminate even more precisely, the *sequence space*, $\mathcal{H}_{\text{seq}}$, associates a trace with the sequence of fault events in the trace, thus preserving the order of faults in addition to their type and number. In a DES, the order in which events take place normally affects the state of the system, so the order of faults may be significant. A natural preference relation on $\mathcal{H}_{\text{seq}}$ is $h \preceq_{\text{seq}} h'$ iff h is a subsequence of h'.

3.2 Properties of Hypothesis Spaces

Next, we introduce some terminology and discuss a number of properties that the hypothesis space may or may not satisfy. These properties will be important to prove termination of different strategies for exploring the hypothesis space, i.e., different diagnosis algorithms.

If $h \preceq h'$, we say that h is a *parent* of h' and that h' is a *child* of h. Note that h is both a parent and a child of itself. If $h \neq h'$ and $\forall h'' \in \mathcal{H},\ h \preceq h'' \preceq h' \Rightarrow (h = h'' \vee h'' = h')$, we say that h is an *immediate parent* of h', and h' is an *immediate child* of h.

Property P1: *The hypothesis space is finite.*

Among the examples discussed above, **P1** holds only for the set space.

Property P2: *Given a non-empty subset $S \subseteq \mathcal{H}$ of hypotheses, the size of the set of minimal elements of S is finite and different from $\emptyset$.*

This is known as $\mathcal{H}$ being a well-partially-ordered set. All the hypothesis spaces presented above satisfy this property. (For the non trivial cases of multisets and sequences, cf. Nash-Williams, 1963).

Property P3: *The size of preferred diagnosis $\min_{\preceq}(\Delta(\mathsf{D}, o, \mathcal{H}))$ is finite.*

Property **P3** is clearly necessary to exhaustively enumerate the preferred diagnosis.

Property P4: *The set of most preferred (i.e., minimal in $\preceq$) hypotheses in $\mathcal{H}$ is finite. The set of immediate children of any hypothesis is finite.*

All hypothesis spaces presented above satisfy **P4**. This property is necessary for the preferred-first strategy (cf. Section 5.1), which iteratively looks at hypotheses starting from the most preferred ones.

Let the *depth* of hypothesis h in $\mathcal{H}$ be defined as the largest number d such that there exists a chain $h_1 \prec h_2 \prec \cdots \prec h_d \prec h$ in $\mathcal{H}$.

Property P5: *Every hypothesis has finite depth.*

P5 does not hold for the multiset space under the refined preference relation, $\preceq_{\text{ms-lex}}$. To see this, assume two faulty events f and g such that $f \prec g$. Let $h = \langle x, y\rangle$ denote that $h(f) = x$ and $h(g) = y$. By definition, $\langle x, y\rangle \preceq_{\text{ms-lex}} \langle x', y'\rangle$ iff $(x < x') \vee (x = x' \wedge y \leq y')$. Because $\langle 0,0\rangle \prec_{\text{ms-lex}} \langle 0,1\rangle \prec_{\text{ms-lex}} \cdots \prec_{\text{ms-lex}} \langle 1,0\rangle$, hypothesis $\langle 1,0\rangle$ has infinite depth. However, **P5** holds for the all the other spaces described above.

Property P6: *For any subset $S \subseteq \mathcal{H}$ and any hypothesis $h \in S$, the subset $\min_{\preceq}(S)$ of minimal hypotheses in S contains a hypothesis $h' \preceq h$.*

Property **P6** is also known as *well-foundedness*. All hypothesis spaces above are well-founded. This is an important property since if the space is not well-founded, the set of preferred candidates could be empty.

Finally, we note the following relations between the properties defined above:

- **P1** implies all other properties.
- **P2** implies **P3**, **P4**, and **P6**.
- **P5** implies **P6**.

4 DIAGNOSIS TESTS

The atomic operation that we use to explore the hypothesis space, and thus to find the preferred diagnosis is the *diagnosis test* (or simply test). A test takes as input the diagnosis problem (i.e., system model and observation) and a subset of hypotheses, S, and asks the question: is there a candidate in S? That is, is there a trace $u \in \mathsf{L}$ such that $obs(u) = o$ and $\delta(u) \in S$? If the answer is yes, the test returns one such hypothesis (or the trace with which it is associated).

Definition 2. *A* diagnosis test *is a tuple $\mathsf{T} = \langle \mathsf{D}, o, S\rangle$ where $\mathsf{D} = \langle \Sigma, \mathsf{L}, obs\rangle$ is the DES, $o \in {\Sigma_o}^*$ the observation, and $S \subseteq \mathcal{H}$ is a set of hypotheses.*

Let $\perp$ be a symbol that does not appear in $\mathcal{H}$. The result $r(\mathsf{T})$ *of test $\mathsf{T} = \langle \mathsf{D}, o, S\rangle$ is a candidate $h \in S$ (called a* test success*), or $\perp$ if no such h exists (called a* test failure*).*

In the following, we will omit DES D and observation o, since they do not vary for a given diagnosis problem, and identify a test simply by the set S.

A *test solver* takes as input a diagnosis test and returns the test result. The test solver can be viewed as searching for a trace u that belongs to the intersection of the following three sets: L, $\mathsf{L}_O = \{u \in \Sigma^* \mid obs(u) = o\}$, and $\mathsf{L}_S = \{u \in \Sigma^* \mid \exists h \in S : \delta(u) = h\}$, and extracting the hypothesis $\delta(u)$, or proving that no such u exists. In practice, however, the inputs given to the solver, including the test set S, are not enumerated but represented symbolically. We describe two different implementations of test solvers in Section 6.

We propose to compute the preferred DES diagnosis by solving a number of carefully choosen tests. The success of this approach depends on i) how efficiently each diagnosis test can be answered, and ii) how many diagnosis tests will be

necessary. Note that, in difference to existing exhaustive diagnosis methods, a test solver does *not* need to compute, even implicitly, the set of traces $Expl = \mathsf{L} \cap \mathsf{L}_O$.

4.1 Diagnosis Tests Used

We now formulate the four different questions, and their corresponding diagnosis tests, which are used to implement the exploration strategies described in the next two sections. Every strategy does not require all of the tests, but these four tests are sufficient to implement all of the strategies.

Given a hypothesis $h \in \mathcal{H}$ and a binary relation $op \in \mathcal{H} \times \mathcal{H}$, we write $S_{op}(h)$ for the set of hypotheses $h' \in \mathcal{H}$ such that h' *op* h.

Question 1. *Is hypothesis h a candidate?*

Test for Question 1:

$$S_{cand}(h) = \{h\}.$$

The next question allows us to determine if a set D of minimal candidates is complete.

Question 2. *Given a set of candidates $D \subseteq \Delta$, does there exist a candidate $h \in \Delta$ such that no candidate $h' \in D$ is preferable to h?*

Test for Question 2:

$$S_{find}(D) = \bigcap_{h' \in D} S_{\not\preceq}(h').$$

$S_{find}(D)$ could equivalently be defined as $\mathcal{H} \setminus (\bigcup_{h' \in D} S_{\succeq}(h'))$. However, as presented at the end of this section, a formulation based on set intersections is very convenient.

The next question is used to "refine" a known diagnosis candidate, i.e., to search for a strictly better candidate.

Question 3. *Is candidate h minimal?*

Test for Question 3:

$$S_{ref}(h) = S_{\prec}(h).$$

The final question relates to the preferred-first strategy, in which we explore $\mathcal{H}$ by going from parent to children. Given a set of hypotheses, H, and a hypothesis h, we want to determine whether the minimal candidates that can be reached through h, i.e., that are (not necessarily immediate) children of h, can also be reached through some other hypothesis in H; if so, h can be ignored.

Question 4. *Given a set of hypotheses $H \subseteq \mathcal{H}$ and a hypothesis h, does there exist a candidate h' such that $h \preceq h'$ and there is no $h'' \in H$ such that $h'' \preceq h$?*

Test for Question 4:

$$S_{ess}(h, H) = S_{\succeq}(h) \cap \bigcap_{h' \in H \setminus \{h\}} S_{\not\preceq}(h').$$

Symbolic Representation of the Test Sets The tests defined above all take the form of an intersection of sets $S_{op}(h)$ where *op* is $\preceq$, $\succeq$ or $\not\preceq$ (e.g., $\{h\} = S_{\preceq}(h) \cap S_{\succeq}(h)$). Therefore, in order to implement these tests, we only need to be able to enforce that a trace u satisfies a conjunction of properties $\delta(u) \in S_{op}(h)$. Two concrete implementations are described in Section 6.

5 EXPLORATION STRATEGIES

In this section, we outline a number of strategies for searching for the preferred diagnosis, and identify for each strategy the conditions on the hypothesis space required to ensure the search terminates with a correct answer. (Proofs of correctness are not complicated, but omitted for lack of space.) A comparison of the different strategies, both theoretically, in terms of the number and types of tests they need to perform (in the worst case), and experimentally, will be presented in Section 7.

5.1 Preferred-First Strategy

A simple approach to diagnosis is a preferred-first approach as proposed by Reiter (1987). This approach will use solely Question 1 and, in its extended version, Question 4.

We name the first algorithm pfs. First, for each minimal hypothesis $h \in \min_{\preceq}(\mathcal{H})$, we test (using Test of Question 1) whether h is a candidate. If h is a candidate, we store it in the result. Otherwise, any immediate child of h could *a priori* be a minimal diagnosis candidate; therefore, we push them in an open list of hypotheses we need to test. One after the other, we pop each hypothesis h' from the open list and we test whether they are candidates (again, Question 1). If h' is a candidate, it can be added to the result; otherwise, its immediate children are inserted in the open list. And so on... It is then necessary to remove non minimal candidates before returning the result, although this is not particularly expensive.

After hypothesis h was proven not to be a candidate, the set of "successors" inserted in the open list could be reduced using a conflict (Reiter, 1987). The generalization of conflicts for any hypothesis space is quite involved and is not presented in this paper.

Theorem 1. *Algorithm* pfs *returns the preferred diagnosis. Furthermore, assuming* **P1** *holds, Algorithm* pfs *always terminates.*

The set space, which is the most common in DES diagnosis, satisfies **P1** and thus ensures termination of pfs.

We now propose an extension to pfs that can be introduced either when **P1** does not hold or in order to reduce the number of diagnosis tests. Question 4 is specifically defined to this end. Let H be the union of the candidates already found and the current open list, and let h be the top element from the open list. If the answer to Question 4 is negative, then any candidate $h'' \succeq h$ is such that $h'' \succeq h'$ also hold for some $h' \in H$. Therefore, exploring h and its children is not essential to achieve completeness since any candidate $h'' \succeq h$ will be reached by exploring h' and its children. Hypothesis h is hence ignored (and its successors are not inserted in the open list). As a side effect, it is unnecessary to remove non minimal candidates. We call pfs+e this algorithm.

Theorem 2. *Algorithm* pfs+e *returns the preferred diagnosis. Furthermore, assuming* **P3**, **P4**, *and* **P5** *hold, Algorithm* pfs+e *always terminates.*

Notice that conflict-based pruning of successors can be seen as an early detection of non-essential hypotheses.

5.2 Preferred-Last Strategy

The preferred-last strategy uses Question 2 and, in its extended version, Question 3.

In this approach, candidates are generated until all minimal candidates have been found. Given a set of candidates already found, in order to make sure the next candidate found is not less preferable than any candidate already found, we use Question 2.

Algorithm pls starts with an empty set D of candidates. It continuously asks Question 2 with set D: if candidate h is generated, h is not less preferable than any candidate in D and it is added to D; on the other hand, if the test of Question 2 returns no candidate, then D contains all minimal candidates and pls may stop. The tests may generate (intermediate) non minimal candidates and they should be eliminated either during the algorithm or at the end.

Theorem 3. *Algorithm* pls *returns the preferred diagnosis. Furthermore, assuming* **P2** *holds, Algorithm* pls *always terminates.*

Algorithm pls randomly finds diagnosis candidates that are not less preferred than the candidates already found. However, there may be many non preferable candidates and the algorithm may need a long time before converging to the solution.

Therefore, we propose an extension of pls where each new found candidate h is "refined" until a minimal candidate $h' \preceq h$ is derived from h. This is somewhat similar to the approach proposed by Feldman *et al.* (2010) for circuits, but notice that we propose to generate all minimal candidates. Refinement is implemented using Test for Question 3. Algorithm pls+r works as follows: starting with an empty set D, it asks Question 2 with D to generate any candidate h; then h is incrementally refined using Question 3 until minimal candidate h' is generated; h' is added to D and Question 2 is asked again. The algorithm stops when Question 2 returns no new candidate.

Theorem 4. *Algorithm* pls+r *returns the preferred diagnosis. Furthermore, assuming* **P3** *and* **P6** *hold, Algorithm* pls+r *always terminates.*

6 IMPLEMENTATION

We now turn to the question of how to practically implement test solvers. We sketch two methods, based on reduction to AI planning and to propositional satisfiability (SAT).

6.1 Reduction to Planning

Each test asks us to find a trace u of the system that has certain properties, or prove that no such trace exists. This problem arises in many other contexts, such as model checking (Clarke *et al.*, 1999) or AI planning (Ghallab *et al.*, 2004), and we can draw upon techniques developed in those fields to solve it. Here, we sketch how tests can be implemented using AI planning.

The classical AI planning problem is, given a DES, an initial state and a goal condition, to find a sequence of transitions from the initial state to one satisfying the goal condition. The DES is represented compactly, by state variables and actions. Each state variable, together with the actions that affect or depend on it, may be viewed as a finite automaton, and the DES is the parallel product (synchronising on action labels) of these automata. For any non-trivial problem, the product is too large to be represented explicitly: all planning algorithms avoid this, in one way or another.

Given a hypothesis set S to be tested, the trace u that we are looking for must generate the correct observation ($obs(u) = o$) and a hypothesis that belongs to the test set ($\delta(u) \in S$). To formulate this as a condition on the goal state, so that we can apply planning methods, we use various encoding tricks. Here we sketch only a few; cf. Haslum & Grastien (2011) for details.

Suppose, for simplicity, that the observation is a sequence of events, $e_o^1, \ldots, e_o^n$. To enforce this, add a new state variable V_o with $n+1$ values, which tracks how far along the sequence we are. Each action that emits event e_o^i is applicable only when $V_o = i - 1$, and changes its value to i. The goal is $V_o = n$. Actions that emit an observable event not in the sequence are simply removed.

A very successful approach to AI planning is heuristic search. A heuristic search-based planner constructs the state-space on-the-fly, using a heuristic to guide the exploration towards parts that appear "promising", i.e., close to a goal state. If the heuristic provides good guidance, and if a goal state exists, the fraction of the state-space that is explored can be very small. However, if no goal state is reachable, the search must explore the entire state space to prove it and the heuristic is of little use. Thus, we can expect failed tests to be computationally much more expensive than successful tests, making this implementation most attractive in combination with the pls strategy.

The pls strategy requires only the test for Question 2, i.e., given a set of candidates D, find a new candidate in $\bigcap_{h \in D} S_{\not\preceq}(h)$. We sketch how this test can be encoded as a planning goal for the multi-set space. A multi-set hypothesis h maps each fault event e_f to the number $h(e_f)$ of times it occurred. For the hypothesis of the trace u to be not dominated by h, i.e., for $h \not\preceq \delta(u)$ to hold, we must have $\delta(u)(e_f) < h(e_f)$ for at least one fault event e_f. Given a candidate set D and a fault event e_f, let $n_f = \max_{h \in D} h(e_f)$ be the maximum count of e_f in any candidate in D. We add a new state variable, V_f, with values $0, \ldots, n$, which tracks the number of occurrences of e_f in trace, but such that $V_f = n$ means "n or more"; that is, each action that emits e_f increments V_f, unless V_f already equals n. The goal that $h \not\preceq \delta(u)$ can then be expressed as the dis-

junction $V_{f_1} < h(e_{f_1}) \vee \ldots \vee V_{f_m} < h(e_{f_m})$. The goal that $h_i \not\preceq \delta(u)$ for each $h_i \in D$ is simply the conjunction of this condition for each h_i.

6.2 SAT-Based Diagnosis

Given a propositional formula Φ defined on a set of Boolean variables, SAT is the problem of finding a variable assignment that makes Φ logically *true*. Finding a trace is reduced to a SAT problem. We assume that there exists an upper-bound n on the number of events in the trace generated by the system. For each event $e \in \Sigma$ and each time step $i \in \{1, \ldots, n\}$, a variable $e@i$ is created that will be assigned to *true* iff the ith event of the trace is e. Formula Φ is defined on this set of variables V to enforce i) that a satisfying assignment corresponds to a trace of L, and ii) that a satisfying assignment corresponds to a trace that generates the observation. More details are given by Grastien *et al.* (2007).

Let u be the trace returned by the SAT solver. We pre-emptively add a number of constraints in Φ so that $\delta(u)$ is in the specified set. We need to enforce three types of sets: $S_{\succeq}(h)$, $S_{\not\preceq}(h)$ (omitted as it is represented as the logical negation of the representation of $S_{\succeq}(h)$), and $S_{\preceq}(h)$.

Set Hypothesis Space

$S_{\succeq}(h)$ For each fault $f \in h$, enforce $f@1 \vee \cdots \vee f@n$.

$S_{\preceq}(h)$ For each fault $f \in \Sigma_f \setminus h$, for each time step $i \in \{1, \ldots, n\}$, enforce $\neg f@i$.

Multiset Hypothesis Space For each fault $f \in \Sigma_f$, Boolean variables $v_{op\ f}$ are defined that are *true* if the number i of occurrences of f in trace u is such that i *op* $h(f)$. This *cardinality constraint* is standard in SAT; the current best implementation that by Bailleux and Boufkhad (2003).

$S_{\succeq}(h)$ Set every $v_{\geq f}$ to *true*.

$S_{\preceq}(h)$ Set every $v_{\leq f}$ to *true*.

Sequence Hypothesis Space Let $seq = [f_1, \ldots, f_k]$ be the sequence associated with hypothesis h; notice that the same fault f may appear several times in seq. For each time step $t \in \{0, \ldots, n\}$ and each index $i \in \{0, \ldots, k\}$, Boolean variable $N_i@t$ is created that is *true* if all faults $f_1, \ldots, f_i$ have been recognized in this order before time step t. The variable semantics can be enforced as follows:

$$N_i@t \leftrightarrow (N_i@(t-1) \vee (N_{i-1}@(t-1) \wedge f_i@t)) .$$

Plus, $N_0@0$ is set to *true* and all other $N_i@0$ are set to *false*. Sequence seq was recognized only if $N_k@n$ is *true*.

We now consider the case where $\delta(u)$ is not a sub sequence of seq, i.e., if it "diverges". For each time step $t \in \{1, \ldots, n\}$ we create a Boolean variable $D@t$ that is true if sequence $\delta(u)$ diverged from seq before or at time step t.

$$D@t \leftrightarrow \quad D@(t-1) \vee (N_i@(t-1) \wedge \neg N_{i+1}@(t-1) \wedge f'@t)$$

where $f' \neq f_{i+1}$. In words, the second disjunct means that f_{i+1} is the next expected fault but f' occurs instead. Variable $D@0$ is set to *false*. If $\delta(u)$ diverges, then $D@n$ is *true*.

In summary, here is how the membership to a set $S_{op}(h)$ is ensured:

$S_{\succeq}(h)$ Set $N_k@n$ to *true*.

$S_{\preceq}(h)$ Set $D@n$ to *false*.

Multiset Hypothesis Space with $\preceq_{\text{ms-lex}}$ To implement the lexical preference order $\preceq_{\text{ms-lex}}$ on the multiset hypothesis space requires only a small modification. Reusing the variables defined above, we have:

$S_{\succeq}(h)$ For each fault $f \in \Sigma_f$, $v_{<f} \rightarrow \bigvee_{f' \prec f}(v_{>f'})$.

$S_{\preceq}(h)$ For each fault $f \in \Sigma_f$, $v_{>f} \rightarrow \bigvee_{f' \prec f}(v_{<f'})$.

7 EVALUATION OF THE APPROACHES

We have presented several exploration strategies and two test implementations. Together, this gives us a large space of possible diagnosis algorithms. We now compare these algorithms and implementations both theoretically and experimentally.

7.1 Theoretical Comparison

Table 1 shows a comparison of the different algorithms in terms of the worst-case number and types of tests each algorithm must solve. In the table, k is the number of successors of a hypothesis, d the depth of minimal candidates, and n the number of minimal candidates.

	pfs	pfs+e	pls	pls+r
Suc.	n	$n \times d$	unb.	unb.
Fail.	unb.	$k \times n \times d$	1	n
Quest.	Q1	Q1 & Q4	Q2	Q2 & Q3

Table 1: Comparison of the algorithms: bounds on the number of successful / failed tests, and questions involved. pls and pls+r both require an unbounded number of tests, though pls+r requires fewer than pls.

We already mentioned the properties necessary to ensure termination of the algorithms. In that respect, pfs is the weakest, as it requires a finite hypothesis space. (Of course, it can still be applied to infinite spaces: it is only the termination *guarantee* that is lost. In the experiments below we use it with the multiset space.) Furthermore, similarly to diagnosis of circuits, if the conflicts used to prune the list of successors are far from minimal, then the number of tests may be very large.

We mentioned that, for the planning-based solver, negative tests are usually harder, making pls and its single failed test an attractive choice. The same is not true of the SAT-based solver. Although the hardest problems are often unsatisfiable ones, not all unsatisfiable problems are hard to solve. Depending on the order in which the SAT solver instantiates the SAT variables, unsatisfiability may be proven very quickly. The encoding of some of the $\delta(u) \in S$ conditions guide the SAT solver quickly to a solution or a failure, while others do not. In particular, the encoding of sets of type $S_{\not\in}(h)$ does not provide the SAT solver with effective guidance, and therefore questions 2 and 4 are usually the most difficult for the SAT solver.

Compared to pfs, the extension pfs+e requires at most twice as many tests. These additional tests correspond to question 4 and many of them are negative; however, these tests allow pruning not only some hypotheses but also their successors, and thus can have a very important positive impact. Finally, algorithm pfs+e has an upper bound on the number of generated tests.

Algorithms pls and pls+r have no such upper bound. However, they can be quite quick at reaching the minimal candidates, in particular if the successful tests return good candidates, i.e., candidates with close to minimal depth. (This is also true for pfs+e which can also use the candidates returned for question 4). The SAT-based test solver usually returns bad candidates; for instance, if the system includes a switch that may be stuck, and if the switch is not operated during the diagnosed period, it will often happen that the returned trace assumes a stuck-fault for the switch. To prevent this behaviour, we analyse the trace returned by the SAT solver and remove *a posteriori* any unnecessary transitions.

Finally, Algorithm pls+r reaches the minimal candidates in fewer tests than pls and using question 3, but at the cost of one negative test per minimal candidate.

7.2 Experiments: Power Grid Alarms

We ran experiments on an alarm processing problem provided by TransGrid, an electricity utility working at the transmission level primarily in New South Wales and in the Australian Capital Territory.

Our goal here is to determine how the alarms generated by the components in the system relate to each other, in order to group the alarms that are clearly symptoms of the same incident and to highlight isolated alarms that could be missed in an otherwise large flow of observation. A fault in our model is defined as an unexplained event. Consider for instance a line protected by four circuit breakers; if all four circuit breakers open at the same time, this can be diagnosed as four unexplained operations of the circuit breakers, or it could be diagnosed as an unexplained fault on the line triggering the operation of all circuit breakers. Because diagnoses with fewer unexplained events are preferred, the multiset hypothesis space is most appropriate.

Pb	#c	#a	#d	pfs+e	pls	pls+r
c-004	2	3	1	0.4	0.3	0.4
c-026	12	11	2	18.5	89.5	23.8
c-076	11	8	2	8.9	7.1	6
c-091	25	16	32	1412	XXX	785
c-111	26	16	8	358	691	173
w-132	16	7	2	14.4	10.5	8.4
w-331	62	44	16	XXX	XXX	545
w-338	42	31	2	820	XXX	166

Table 2: Runtime for some selected power grid problems. # c measures the number of components, # a the number of alarms, and # d the number of minimal candidates; XXX means the 30 min time limit was reached.

The whole network comprises over $10,000$ components. Given a log of alarms, we can easily restrict the set of components considered to those that can play a role. Consequently, we have a different number of components for each problem, from 2 to 104 for the most difficult problem. The number of observed events ranges from 2 to 146.

The problems were solved using Algorithms pfs+e, pls, and pls+r, and a SAT-based test solver powered by MiniSAT (Eén and Sörensson, 2003).

Out of 129 diagnostic problems, we present the results for several representative problems on Table 2. In this setting, algorithm pls+r generally performs best. The counter-example of c-026 corresponds to an occurrence where the refinement was particularly unlucky, leading to 13 SAT calls for pls+r against 10 calls for pfs (and a depth of two for both candidates) and 47 for pls; on the other hand, candidates of w-338 have depth 24 and 25; this problem required 67 SAT calls for pfs+e, and only 13 calls for pls+r while pls reached the time limit of 30 min after finding 123 (non minimal) candidates, 20 of which were minimal with respect to each other.

In the current implementation, all calls to the SAT solver are performed independently. In particular, the solver does not reuse anything learnt on a previous problem, for instance the fact that it was proved that a particular event did not take place. When the number of SAT calls increases, it would become very beneficial either to reuse this knowledge or to run a non expensive pre-process on the diagnosis problem to extract useful information that could be included for all tests.

We also ran experiments on a UAV benchmark (Haslum and Grastien 2011; details omitted for lack of space). On this problem set, the planning-based implementation of Algorithm pls performs better than all SAT-based implementations, probably due to the very sequences of both observable and unobservable events that characterise these problems.

8 CONCLUSION

We proposed a new approach to DES diagnosis where we explore the hypothesis space rather than using a monitoring-based approach. The advantage of this approach is that it avoids following

all traces of the model that can explain the observation. Instead, only traces associated with some hypotheses are considered which automatically reduces the complexity of diagnosis.

This different approach opens new research perspectives. First, we used solvers that were not specifically designed to answer the tests required by our algorithms. Either using special purpose solvers or introducing application-specific changes into existing SAT or planning solvers (in the line of (Rintanen, 2010)) could have dramatic impact on runtime.

Another opportunity is the generation of conflicts. We suspect an approach very similar to the one used for diagnosis of circuits can be developed. The idea would be to define set S as the set of hypotheses that satisfy a conjunction of properties; when a test fails, it should be possible to determine a subset of properties that are sufficient to make the test unsatisfiable. This subset represents a conflict and can be used to generate a subset of successors. We suspect this operation is a special case of ignorance of non essential hypotheses.

ACKNOWLEDGMENTS

We thank TransGrid for their permission to use the data. This work was supported by ARC project DP0985532. 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.

REFERENCES

(Aghasaryan *et al.*, 1998) Ar. Aghasaryan, É. Fabre, Al. Benveniste, R. Boubour, and Cl. Jard. Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri Nets. *Journal of Discrete Event Dynamical Systems*, 8(2):203–231, 1998.

(Bailleux and Boufkhad, 2003) O. Bailleux and Ya. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Ninth International Conference on Principles and Practice of Constraint Programming (CP-03)*, pages 108–122, 2003.

(Cassandras and Lafortune, 1999) Chr. Cassandras and St. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.

(Clarke *et al.*, 1999) E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

(Cordier *et al.*, 1997) M.-O. Cordier, S. Thiébaux, O. Jehl, and J.-P. Krivine. Supply restoration in power distribution systems: A reference problem in diagnosis and reconfiguration. In *Eighth International Workshop on Principles of Diagnosis (DX-97)*, pages 37–44, 1997.

(Cordier *et al.*, 2006) M.-O. Cordier, L. Travé-Massuyès, and X. Pucel. Comparing diagnosability in continuous and discrete-event systems. In *Seventeenth International Workshop on Principles of Diagnosis (DX-06)*, pages 55–60, 2006.

(Dague, 1994) Ph. Dague. Model-based diagnosis of analog electronic circuits. *Annals of Mathematics and Artificial Intelligence*, 11:439–492, 1994.

(Darwiche, 1998) Ad. Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

(Eén and Sörensson, 2003) N. Eén and N. Sörensson. An extensible SAT-solver. In *Sixth Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, pages 333–336, 2003.

(Feldman *et al.*, 2010) A. Feldman, Gr. Provan, and Ar. van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, 38:371–413, 2010.

(Ghallab *et al.*, 2004) M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers, 2004. ISBN: 1-55860-856-7.

(Grastien *et al.*, 2007) Al. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Modeling and solving diagnosis of discrete-event systems via satisfiability. In *Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, 2007.

(Haslum and Grastien, 2011) P. Haslum and A. Grastien. Diagnosis as planning: Two case studies. In *Proc. ICAPS'11 Workshop on Scheduling and Planning Applications*, pages 37–44, 2011. `http://icaps11.icaps-conference.org/workshops/spark.html`.

(Nash-Williams, 1963) Cr. Nash-Williams. On well-quasi-ordering finite trees. *Mathematical Proceedings of The Cambridge Philosophical Society*, 59(4):833–835, 1963.

(Pulido and Alonso González, 2002) B. Pulido and C. Alonso González. Possible conflicts, ARRs, and conflicts. In *Thirteenth International Workshop on Principles of Diagnosis (DX-02)*, 2002.

(Reiter, 1987) R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

(Rintanen, 2010) J. Rintanen. Heuristic planning with SAT: beyond strict depth-first search. In *23rd Australasian Joint Conference on Artificial Intelligence (AJCAI-10)*, 2010.

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

(Sohrabi *et al.*, 2010) Sh. Sohrabi, J. Baier, and Sh. McIlraith. Diagnosis as planning revisited: an abridged report. In *21st International Workshop on Principles of Diagnosis (DX-10)*, pages 143–150, 2010.

(Zanella and Lamperti, 2003) M. Zanella and Gi. Lamperti. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.