Computing Multiple Minimal Diagnoses

Alexander Feldman[1] and Gregory Provan[2] and Arjan van Gemund[1]

[1] *Delft University of Technology, Faculty of Electrical Engineering Mathematics and Computer Science*
Mekelweg 4, 2628 CD, Delft, The Netherlands, {a.b.feldman,a.j.c.vangemund}@tudelft.nl
[2] *University College Cork, Department of Computer Science*
College Road, Cork, Ireland, g.provan@cs.ucc.ie

ABSTRACT

Existing research in Model-Based Diagnosis (MBD) primarily concerns computation of a single (possibly multiple-fault) diagnostic candidate. This is unrealistic, as often multiple candidates cannot be discerned given a system description and an observation vector. It is also computationally more difficult to compute multiple minimal-cardinality diagnoses, as opposed to a single diagnosis. In this paper we analyze the theoretical and practical aspects of computing multiple minimal-cardinality diagnoses. We propose an algorithm, named SAFARI, which solves the computational complexity problem by trading-off completeness for efficiency. Our algorithm has the desirable property of computing multiple-cardinality diagnoses with probability which is negatively exponential to the cardinality of the minimal-cardinality diagnoses. We also empirically confirm the theoretical results with experiments on a benchmark of 74XXX/ISCAS85 combinational circuits. The efficiency of the algorithm is evaluated in terms of metrics, and the results are compared to other MBD algorithms participating in the First International Diagnostic Competition (DXC'09). The results from the competition support our theoretical prediction that computing all minimal-cardinality diagnoses maximizes the DXC'09 utility metric. The results also show at least an order-of-magnitude speedup and an order-of-magnitude decrease in memory consumption while computing multiple minimal diagnoses of optimality similar to competing algorithms.

1 INTRODUCTION

LYDIA (Feldman *et al.*, 2006) is a declarative modeling language specifically developed for Model-Based Diagnosis (MBD) (de Kleer and Williams, 1987). The accompanying toolset currently comprises a number of diagnostic engines and a simulator tool. SAFARI (Feldman *et al.*, 2008a) (StochAstic Fault diagnosis AlgoRIthm), is a LYDIA module that computes minimal diagnoses while sacrificing guarantees of optimality, but for diagnostic systems in which faults are described in terms of an arbitrary deviation from nominal behavior, SAFARI can compute diagnoses several orders of magnitude faster than existing state-of-the-art algorithms.

SAFARI competed in the synthetic track of the First International Diagnostic Competition (DXC'09) against the diagnostic algorithms NGDE (de Kleer, 2009) and RODON (Bunus *et al.*, 2009). The CPU and memory performance of SAFARI were an order-of-magnitude better than the competing algorithms despite the fact that NGDE and RODON performed better than other MBD algorithms like CDA* (Williams and Ragno, 2007) or HA* (Feldman and van Gemund, 2006).

The contributions of this paper are as follows. (1) We introduce an algorithm for computing multiple minimal-cardinality diagnoses. (2) We show that this algorithm computes minimal diagnoses with probability which is negatively exponential to the cardinality of the diagnoses. (3) We define performance metrics and compare SAFARI to other algorithms from DXC'09.

This paper is of great practical significance as it bridges algorithmic and theoretical problems (computation of multiple minimal diagnoses) and practice (DXC'09). We believe that MBD algorithms which can compute diagnoses in systems comprising of thousands of variables and constraints would allow easier modeling, automatic generation of models from design specifications, proving model properties, etc.

This paper is organized as follows. Section 2 contains basic definitions. Section 3 describes the SAFARI algorithm. Section 4 explains properties of SAFARI in computing multiple minimal diagnoses and presents a Monte Carlo algorithm for simulating SAFARI. Section 5 introduces the DXC'09 metrics and links them to the theoretical claims made earlier. Section 6 contains experimental results from DXC'09.

2 TECHNICAL BACKGROUND

This section formalizes some standard MBD notions, and explains, on an intuitive level, the SAFARI inference algorithm. It is important to note that SAFARI performs inference in a manner that is different to "standard" MBD algorithms, such as GDE (de Kleer and Williams, 1987), in that it interleaves stochastic search with SAT consistency-checking of potential diagnoses, as we will explain.

2.1 Diagnosis and Minimal Diagnosis

We adopt the traditional diagnostic definitions (de Kleer and Williams, 1987), except that we use propositional logic terms (conjunctions of literals) instead of sets of failing components.

Central to MBD, a *model* of an artifact is represented as a propositional $\mathbf{Wff}$ over some set of variables. Discerning two subsets of these variables as *assumable* and *observable*[1] variables gives us a diagnostic system.

Definition 1 (Diagnostic System)**.** A diagnostic system DS is defined as the triple $\mathrm{DS} = \langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$, where SD is a propositional theory over a set of variables V, $\mathrm{COMPS} \subseteq V$, $\mathrm{OBS} \subseteq V$, COMPS is the set of assumables, and OBS is the set of observables.

Throughout this paper we assume that $\mathrm{OBS} \cap \mathrm{COMPS} = \emptyset$ and $\mathrm{SD} \not\models \perp$. Although SAFARI delivers good results for a larger class of diagnostic models, this paper focuses on the well-known weak-fault models[2].

Definition 2 (Weak-Fault Model)**.** A diagnostic system $\mathrm{DS} = \langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$ belongs to the class $\mathbf{WFM}$ iff SD is in the form $(h_1 \Rightarrow F_1) \wedge \ldots \wedge (h_n \Rightarrow F_n)$ such that $1 \leq i, j \leq n$, $\{h_i\} \subseteq \mathrm{COMPS}$, $F_j \in \mathbf{Wff}$, and none of h_i appears in F_j.

Note the conventional selection of the sign of the "health" variables $h_1, h_2, \ldots, h_n$. Other authors use "ab" for abnormal or "ok" for healthy. Weak-fault models are sometimes referred to as models with *ignorance of abnormal behavior* (de Kleer *et al.*, 1992), or *implicit fault systems*. The traditional query in MBD computes terms of assumable variables which are explanations for the system description and an observation.

Definition 3 (Health Assignment)**.** Given a diagnostic system $\mathrm{DS} = \langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$, an assignment HA to all variables in COMPS is defined as a health assignment.

A health assignment HA is a conjunction of propositional literals. In some cases it is convenient to use the set of negative or positive literals in HA. These two sets are denoted as $Lit^-(\mathrm{HA})$ and $Lit^+(\mathrm{HA})$, respectively.

What follows is a formal definition of consistency-based diagnosis.

Definition 4 (Diagnosis)**.** Given a diagnostic system $\mathrm{DS} = \langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$, an observation α over some variables in OBS, and a health assignment ω, ω is a diagnosis iff $\mathrm{SD} \wedge \alpha \wedge \omega \not\models \perp$.

[1]In the MBD literature the assumable variables are also referred to as "component", "failure-mode", or "health" variables. Observable variables are also called "measurable", or "control" variables.

[2]DXC'09 provides the topology and nominal behavior of the ISCAS85 circuits only. Participants may assume "stuck-at" behavior but we have used weak-fault models only.

In the MBD literature, a range of types of "preferred" diagnosis has been proposed. This turns the MBD problem into an optimization problem. In the following definition we consider the common subset-ordering.

Definition 5 (Minimal Diagnosis)**.** A diagnosis $\omega^{\subseteq}$ is minimal if no diagnosis $\tilde{\omega}^{\subseteq}$ exists such that $Lit^-(\tilde{\omega}^{\subseteq}) \subset Lit^-(\omega^{\subseteq})$.

The set of all minimal diagnoses characterizes all diagnoses given a weak-fault model, but that does not hold in general (de Kleer *et al.*, 1992). With no restrictions on the model, faulty components may "exonerate" each other. In the latter case a health assignment containing a proper superset of the negative literals of a (minimal) diagnosis may not to be a diagnosis.

Definition 6 (Number of Minimal Diagnoses)**.** Let the set $\Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)$ contain all minimal diagnoses of a system description SD and an observation α. The number of minimal diagnoses, denoted as $|\Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)|$, is defined as the cardinality of $\Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)$.

Diagnosis cardinality gives us another partial ordering: a diagnosis is defined as *minimal cardinality* iff it minimizes the number of negative literals. The cardinality of a diagnosis, denoted as $|\omega|$, is defined as the number of negative literals in ω.

Definition 7 (Minimal-Cardinality Diagnosis)**.** A diagnosis $\omega^{\leq}$ is defined as minimal-cardinality if no diagnosis $\tilde{\omega}^{\leq}$ exists such that $|\tilde{\omega}^{\leq}| < |\omega^{\leq}|$.

In this text we may refer to a minimal diagnosis (Def. 5) as to a subset-minimal diagnosis. A minimal-cardinality diagnosis, though, will always spell the word *cardinality*.

Definition 8 (Number of Minimal-Cardinality Diagnoses)**.** Let the set $\Omega^{\leq}(\mathrm{SD} \wedge \alpha)$ contain all minimal-cardinality diagnoses of a system description SD and an observation α. The number of minimal-cardinality diagnoses, denoted as $|\Omega^{\leq}(\mathrm{SD} \wedge \alpha)|$, is defined as the cardinality of $\Omega^{\leq}(\mathrm{SD} \wedge \alpha)$.

A minimal cardinality diagnosis is a minimal diagnosis ($\Omega^{\leq}(\mathrm{SD} \wedge \alpha) \subseteq \Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)$), but the opposite does not hold. There are minimal diagnoses which are not minimal cardinality diagnoses.

2.2 Diagnostic Inference: "Traditional" versus SAFARI Methodologies

This section provides an intuitive comparison of SAFARI with "standard" MBD algorithms, in order to clarify our novel algorithm, which will be explained in more detail in the following section. SAFARI performs inference in a manner that is different than "standard" MBD algorithms. In the following, we first summarize the standard MBD approach, and contrast it with that of SAFARI.

Given an observation OBS denoting an abnormal condition (or symptom), a standard MBD algorithm A performs diagnosis in a two-step process. In the first step, A computes the conflicts for OBS, i.e., a conflict is a set of components which cannot all be operating properly given a symptom. We denote a conflict ζ as an assignment to a subset of health variables.

Definition 9 (Conflict)**.** Given a diagnostic system $\mathrm{DS} = \langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$, and an observation α over some variables in OBS, ζ is a conflict iff $\mathrm{SD} \wedge \alpha \wedge \zeta \models \perp$.

Most MBD approaches will compute minimal conflicts, i.e., conflicts which are minimal with respect to some preference criterion ϕ. Both subset- and cardinality-minimal conflicts have been studied in the literature. Conflicts are typically computed using constraint-propagation methods (de Kleer and Williams, 1987). The minimal conflicts can be computed by, for example, the ATMS (de Kleer, 1986).

In the second step, a standard MBD algorithm A computes, from a set of minimal conflicts ζ, a diagnosis (or preferred diagnosis, using some preference function ϕ). A diagnosis is a health assignment to each system component, as defined before. Diagnoses (or minimal diagnoses with respect to ϕ) are computed using some logic-based consistency-checking mechanism, such as a SAT solver (McAllester, 1990) or through computing a minimal hitting set of the minimal conflicts (de Kleer and Williams, 1987).

In contrast, SAFARI performs diagnostic inference as follows. SAFARI avoids the conflict-analysis phase by (1) guessing an initial diagnosis ω given an observation α, and then (2) trying to reduce the cardinality of this initial diagnosis ω through flipping the values of some health variables in ω from faulty to healthy, performing a consistency-check after each flip. SAFARI uses an incomplete SAT-solver, BCP, for each consistency-check (McAllester, 1990). This randomized search, in conjunction with the computationally efficient consistency-checking, is what distinguishes SAFARI from traditional algorithms, and also what gives it its computational advantages over traditional algorithms.

3 GREEDY STOCHASTIC COMPUTATION OF DIAGNOSES

This section presents a more formal specification of SAFARI, an algorithm for computing multiple-fault diagnoses using stochastic search.

3.1 The SAFARI Algorithm

As described earlier, SAFARI uses a two-step diagnostic process. Step 1 involves randomly choosing candidates. Step 2 attempts to minimize the fault cardinality of these candidates.

Algorithm 1 shows the top-level pseudocode for SAFARI. Step 1 takes place in line 3 of Algorithm 1; the remainder of the pseudocode of Algorithm 1 performs Step 2.

Algorithm 1 uses a number of utility functions, which we briefly review. The IMPROVEDIAGNOSIS subroutine takes a term as an argument and changes the sign of a random negative literal. If there are no negative literals, the function returns the original argument. The implementation of RANDOMDIAGNOSIS uses a modified Davis-Putnam-Logemann-Loveland (DPLL) solver (Davis *et al.*, 1962) returning a random SAT solution of $\mathrm{SD} \wedge \alpha$.

Similar to deterministic methods for MBD, SAFARI uses a SAT-based procedure for checking the consistency of $\mathrm{SD} \wedge \alpha \wedge \omega$. Because $\mathrm{SD} \wedge \alpha$ does not change during the search, the incremental nature of the Logic-Based Truth Maintenance System (LTMS) assumption checking (McAllester, 1990) greatly improves the search efficiency. The implementation of SAFARI uses LTMS which is based on Boolean Constraint Propagation (BCP), to check for inconsistencies. If a candidate is consistent, a subsequent DPLL check is invoked for completeness.

The randomized search process performed by SAFARI has two parameters, M and N. There are N independent searches that start from randomly generated starting points. The algorithm tries to improve the cardinality of the initial diagnoses (while preserving their consistency) by randomly "flipping" fault literals. The change of a sign of literal is done in one direction only: from faulty to healthy.

Algorithm 1 SAFARI: A greedy stochastic hill climbing algorithm for approximating the set of minimal diagnoses.

```
 1: function SAFARI($\mathrm{DS}, \alpha, M, N$) returns a trie
        inputs: DS = $\langle \mathrm{SD}, \mathrm{COMPS}, \mathrm{OBS} \rangle$
                    a diagnostic system
                $\alpha$, term, observation
                $M$, integer, climb restart limit
                $N$, integer, number of tries
        local variables: $m, n$, integers
                         $\omega, \omega'$, terms
                         $R$, set of terms, result
 2:     for $n = 1, 2, \ldots, N$ do
 3:         $\omega \leftarrow$ RANDOMDIAGNOSIS($\mathrm{SD}, \alpha$)
 4:         $m \leftarrow 0$
 5:         while $m < M$ do
 6:             $\omega' \leftarrow$ IMPROVEDIAGNOSIS($\omega$)
 7:             if $\mathrm{SD} \wedge \alpha \wedge \omega' \not\models \perp$ then
 8:                 $\omega \leftarrow \omega'$
 9:                 $m \leftarrow 0$
10:             else
11:                 $m \leftarrow m + 1$
12:             end if
13:         end while
14:         unless ISSUBSUMED($R, \omega$) then
15:             ADDTOTRIE($R, \omega$)
16:             REMOVESUBSUMED($R, \omega$)
17:         end unless
18:     end for
19:     return $R$
20: end function
```

Each attempt to find a minimal diagnosis terminates after M unsuccessful attempts to "improve" the current diagnosis stored in ω. Thus, increasing M will lead to a better exploitation of the search space and, possibly, to diagnoses of lower cardinality, while decreasing it will improve the overall speed of the algorithm.

It is possible that two diagnostic searches may result in the same minimal diagnosis. To prevent this, we store the generated diagnoses in a trie R (Forbus and de Kleer, 1993), from which it is straightforward to extract the resulting diagnoses by recursively visiting its nodes. A diagnosis ω is added to the trie R by the function ADDTOTRIE, iff no subsuming diagnosis is contained in R (the IsSubsumed subroutine checks on that condition). After adding a diagnosis ω to the resulting trie R, all diagnoses contained in R and subsumed by ω are removed by a call to RemoveSubsumed.

3.2 A Simple Example

We will use the Boolean circuit shown in Fig. 1 as a running example for illustrating Safari. The subtractor, shown there, consists of seven components: an inverter, two or-gates, two xor-gates, and two and-gates. The expression $h \Rightarrow (o \Leftrightarrow \neg i)$ models the normative (healthy) behavior of an inverter, where the variables i, o, and h represent input, output and health respectively. Similarly, an and-gate is modeled as $h \Rightarrow (o \Leftrightarrow i_1 \wedge i_2)$ and an or-gate by $h \Rightarrow (o \Leftrightarrow i_1 \vee i_2)$. Finally, an xor-gate is specified as $h \Rightarrow [o \Leftrightarrow \neg (i_1 \Leftrightarrow i_2)]$.

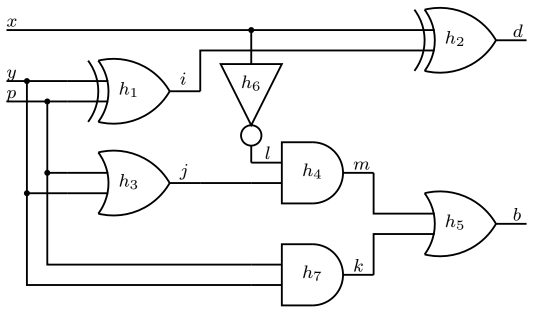

Figure 1: A subtractor circuit

The above propositional formulae are copied for each gate in Fig. 1 and their variables renamed in such a way as to properly connect the circuit and disambiguate the assumables, thus obtaining a propositional formula for the Boolean subtractor, given by:

$$\mathrm{SD} = \begin{cases} h_1 \Rightarrow [i \Leftrightarrow \neg (y \Leftrightarrow p)] \\ h_2 \Rightarrow [d \Leftrightarrow \neg (x \Leftrightarrow i)] \\ h_3 \Rightarrow (j \Leftrightarrow y \vee p) \\ h_4 \Rightarrow (m \Leftrightarrow l \wedge j) \\ h_5 \Rightarrow (b \Leftrightarrow m \vee k) \\ h_6 \Rightarrow (x \Leftrightarrow \neg l) \\ h_7 \Rightarrow (k \Leftrightarrow y \wedge p) \end{cases} \tag{1}$$

The set of assumables is $\mathrm{COMPS} = \{h_1, h_2, \ldots, h_7\}$ and the set of observable variables is $\mathrm{OBS} = \{x, y, p, d, b\}$.

Diagnostic inference with Safari proceeds as follows. In step 1, the stochastic diagnostic search for the subtractor example will start from a random quintuple candidate[3]. In this particular version of our algorithm, once a component is marked as healthy, it cannot be changed back to faulty. To compensate for that, we perform multiple restarts from a random candidate. In our subtractor example and for $\alpha_3 = x \wedge y \wedge p \wedge \neg d \wedge \neg b$, if $h_1 \wedge h_2$ is in an initial "guessed" candidate, it will prove inconsistent with $\mathrm{SD} \wedge \alpha_3$ and another quintuple fault candidate will be guessed.

[3]Feldman *et al* (2008b) describe a method for determining the initial candidates.

Assume that this second candidate is $\omega_6 = \neg h_1 \wedge \neg h_2 \wedge h_3 \wedge \neg h_4 \wedge \neg h_5 \wedge h_6 \wedge \neg h_7$. Clearly, $\mathrm{SD} \wedge \alpha_3 \wedge \omega_6 \not\models \perp$. The search algorithm may next try to improve the diagnosis by "flipping" the sign of h_7. The candidate $\omega_7 = \neg h_1 \wedge \neg h_2 \wedge h_3 \wedge \neg h_4 \wedge \neg h_5 \wedge h_6 \wedge h_7$ is a valid quadruple fault diagnosis and it can be improved twice more by "flipping" h_2 and h_5. This gives us the final double-fault $\omega_8 = \neg h_1 \wedge h_2 \wedge h_3 \wedge \neg h_4 \wedge h_5 \wedge h_6 \wedge h_7$. The actual algorithm is somewhat more involved as during the variable flipping it is normal to find inconsistencies. Instead of restarting, it will simply discard these inconsistent candidates until some termination criterion is satisfied.

Intuitively, from our example, due to the large number of double fault diagnoses explaining the same observation, it is not difficult to randomly guess sequences of variables which need to be false in order to explain the observation.

4 PROPERTIES OF SAFARI IN COMPUTING MULTIPLE DIAGNOSES

We have shown (Feldman *et al.*, 2008a) that for $M = |\mathrm{COMPS}|$ and $\mathrm{SD} \in \mathbf{WFM}$, Safari is guaranteed to find N (not necessarily distinct) minimal diagnoses in each run.

Consider a system description SD ($\mathrm{SD} \in \mathbf{WFM}$) and an observation α. The number of minimal diagnoses $|\Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)|$ can be exponential in $|\mathrm{COMPS}|$. Furthermore, in practice, diagnosticians are interested in sampling from the set of minimal-cardinality diagnoses $\Omega^{\leq}(\mathrm{SD} \wedge \alpha)$ (recall that $\Omega^{\leq}(\mathrm{SD} \wedge \alpha) \subseteq \Omega^{\subseteq}(\mathrm{SD} \wedge \alpha)$) as the minimal-cardinality diagnoses cover a significant part of the *a posteriori* diagnosis probability space (de Kleer, 1990). In what follows, we will see that Safari is very well suited for that task.

Theorem 1. *The probability of* Safari*, configured with $M = |\mathrm{COMPS}|$, computing a diagnosis of cardinality $|\omega|$ in a system with $|\mathrm{COMPS}|$ component variables approaches $O\left(|\mathrm{COMPS}|^{|\omega|}\right)$ for $|\mathrm{COMPS}|/|\omega| \to \infty$.*

Proof (Sketch). For notational brevity we will denote $\gamma = |\mathrm{COMPS}|$. Assume a minimal diagnosis of cardinality $|\omega|$ exists. From the Minimal Diagnosis Guarantee[4] it follows that Safari configured with $M = \gamma$ is guaranteed to compute minimal diagnoses. Starting from the "all faulty" assignment, consider a step k in "improving" the diagnosis cardinality. If state k contains more than one diagnosis, then at state $k+1$, Safari will either (1) flip a literal belonging to this diagnosis (note that a literal may belong to more than one diagnosis) and subsequently prevent Safari of reaching this diagnosis or (2) flip a literal belonging to a diagnosis which has already been invalidated (i.e., one or more of its literals have been flipped at an earlier step).

The probability that a solution of cardinality $|\omega|$ "survives" a flip at iteration k (i.e., is not invalidated)

[4]Cf. Proposition 1 in the related paper of Feldman *et al.* (2008a).

is:

$$p\left(k\right) = 1 - \frac{|\omega|}{\gamma - k} = \frac{\gamma - |\omega| - k}{\gamma - k} \tag{2}$$

The probability that a diagnosis ω "survives" until it is returned by the algorithm:

$$f(\gamma - |\omega| - 1) = \prod_{i=0}^{\gamma-|\omega|-1} p(i) = \tag{3}$$

$$= \prod_{i=0}^{\gamma-|\omega|-1} \frac{\gamma - |\omega| - i}{\gamma - i} \tag{4}$$

Rewriting the right hand side of Eq. (3) gives us:

$$f(\gamma - |\omega| - 1) = \frac{(\gamma - |\omega|)!}{(|\omega| + 1)(|\omega| + 2) \cdots \gamma} = \tag{5}$$

$$= \frac{|\omega|!(\gamma - |\omega|)!}{\gamma!} \tag{6}$$

Since

$$\frac{(\gamma - |\omega|)!}{\gamma!} = \frac{1}{(\gamma - |\omega| + 1)(\gamma - |\omega| + 2) \cdots \gamma} \tag{7}$$

it holds that

$$\lim_{\gamma/|\omega| \to \infty} \frac{(\gamma - |\omega|)!}{\gamma!} = \gamma^{-|\omega|} \tag{8}$$

As a result, for small $|\omega|$ relative to γ,

$$f(\gamma - |\omega| - 1) = |\omega|! O\left(\gamma^{|\omega|}\right) \tag{9}$$

which gives us the above theorem. □

The distribution $h_i(|\omega|)$ of the cardinalities of the minimal diagnoses in $\Omega^{\subseteq}(\text{SD} \wedge \alpha)$ depends on the topology of SD and on α; i.e., we can create SD and α having any $h_i(|\omega|)$. We denote the cardinality distribution of the minimal diagnoses computed by SAFARI as $h(|\omega|)$.

If we have N_c diagnoses of minimal-cardinality c one wants to know when to stop (i.e., what is the probability of missing out a diagnosis). The number of SAFARI runs for computing yet another diagnosis of cardinality c, grows according to a first-order function (the probability of computing a new diagnosis decreases geometrically). After computing some constant number of diagnoses we can simply fit the data with a first-order function and estimate its parameters (in this case N_c is the initial value, the decay constant λ_c is of less interest) using a simple regression technique that also allows us to compute the confidence interval.

Theorem 1 gives us a termination criterion for SAFARI which can be used for enumerating and counting minimal-cardinality diagnoses. Instead of running SAFARI with a fixed N it is sufficient to compute the area under the output distribution function $\sum h$. This value will converge to a single value, hence we can terminate SAFARI after the change of $\sum h$ drops below a fixed threshold. Note that SAFARI is efficient in enumerating the minimal-cardinality diagnoses as they are computed with a probability that is exponentially higher than that of the probability of computing minimal diagnoses of higher-cardinality.

Corollary 1. SAFARI *computes diagnoses of equal cardinality with equal probability.*

Proof (Sketch). From Theorem 1 it follows that the probability of success f of SAFARI in computing a specific diagnosis ω depends only on $|\omega|$ and not on the actual composition of ω. □

The above corollary gives us a simple termination criterion for SAFARI in the cases when all minimal diagnoses are also minimal-cardinality diagnoses; it can be proven that in this case all minimal-cardinality diagnoses are computed with the same probability.

We will see that, given an input cardinality distribution $h_i(|\omega|)$, SAFARI produces an output distribution $h(|\omega|)$ that is highly skewed to the right due to Theorem 1. To facilitate the study of how SAFARI transforms $h_i(|\omega|)$ into $h(|\omega|)$ we will use a Monte Carlo simulation of SAFARI. The advantage is that the Monte Carlo simulation is much simpler for analyzing the run-time behavior of SAFARI than studying the algorithm itself.

Algorithm 2 Monte Carlo simulation of SAFARI.

```
1: function SAFARISIMULATE($\Omega^{\subseteq}, N$) returns a cardinality distribution
       inputs: $\Omega^{\subseteq}$, a set of minimal diagnoses
               $N$, integer, number of tries
       local variables: $h_i, h$, vectors, card. distr.
                        $b$, vector, fault distribution
                        $n, i, c$, integers
2:     $h_i \leftarrow$ CARDINALITYDISTRIBUTION($\Omega^{\subseteq}$)
3:     for $n \leftarrow 1, 2, \ldots, N$ do
4:         for $c \leftarrow 1, 2, \ldots, |h_i|$ do
5:             $b[c] \leftarrow c \cdot h_i[c]$
6:         end for
7:         for $i \leftarrow 1, 2, \ldots, |\Omega^{\subseteq}|$ do
8:             $c \leftarrow$ DISCRRND$^{-1}$ $(b/\sum b)$
9:             $b[c] \leftarrow b[c] - c$
10:        end for
11:        $h[c] \leftarrow h[c] + 1$
12:    end for
13:    return $h$
14: end function
```

Algorithm 2 simulates which diagnoses from the input set of minimal diagnoses Ω are "reached" by SAFARI in N tries. The auxiliary subroutine CARDINALITYDISTRIBUTION computes the input distribution h_i by iterating over all diagnoses in $\Omega_{\subseteq}$. We store the input cardinality distribution h_i and the resulting cardinality distribution h in vectors (note the vector sums in lines 7 and 8 and the division of a vector by scalar in line 8).

The outermost loop of Alg. 2 (lines 3 – 12) simulates the N runs of SAFARI. This is done by computing and updating an auxiliary vector b, which contains the distribution of the component variables in $\Omega^{\subseteq}$ according to the cardinalities of the diagnoses these variables belong to. Initially, b is initialized with the number of literals in single faults in position 1, the number of literals in double faults in position 2 (for example if there are three double faults in h_i, $b[2] = 6$), etc. This is done in lines 4 – 6 of Alg. 2. Note that in order to simplify the simulation algorithm, we assume that diagnoses do not share literals. This restriction can be easily dropped by counting all the assumables in the input $\Omega^{\subseteq}$. The latter assumption does not change the results of this section as the fact that two diagnoses ω_1 and ω_2 share literals does not change the individual probability of ω_1 or ω_2 being "hit" or "invalidated".

Lines 7 – 10 simulate the process of the actual bit flipping of SAFARI. At each step the simulation draws a random literal from the probability distribution function (*pdf*) $\frac{b}{\sum b}$; this is done by the DISCRRND^{-1} function in line 8. Each bit flip "invalidates" a diagnosis from the set $\Omega^{\subseteq}$, i.e., a diagnosis of cardinality c cannot be reached by SAFARI. After a diagnosis has been "invalidated", the vector b is updated, for example, if the simulation "invalidates" a quadruple fault, $b[4] = b[4] - 4$ (line 9). Note that the number of iterations in the loop in lines 7 – 10 equals the number of diagnoses in $\Omega^{\subseteq}$. As a result after terminating this loop, the value of the integer variable c is equal to the cardinality of the *last* "invalidated" diagnosis. The latter is the diagnosis which SAFARI computes in this run. What remains is to update the resulting pdf with the right cardinality (line 11).

The simulation in Alg. 2 links the distribution of the actual diagnoses in $\Omega^{\subseteq}$ to the distribution of the cardinalities of the diagnoses returned by SAFARI. As $\Omega^{\subseteq}$ can be arbitrarily set, we will apply Alg. 2 to a range of typical input distributions. The results of the simulation as well as the results of running SAFARI on synthetic problems with the same input distributions are shown in Fig. 2.

Fig. 2 shows (1) that Alg. 2 predicts the actual behavior of SAFARI (compare the second and third column of plots), and (2) that SAFARI computes diagnoses of small cardinality in agreement with Theorem 1. The only case when the output distribution is not a steep exponential is when the cardinalities in the set of the input minimal diagnoses grow exponentially. Table 1 summarizes the parameters of exponential fits for the input cardinality distributions shown in Fig. 2 (a is the initial (zero) cardinality, λ is the decay constant, and R^2 is the coefficient of determination).

Input Distribution	a	λ	R^2
Uniform	576	-0.44	1
Normal	423	-0.34	0.99
Exponential	69 470	-4.26	1
Reverse Exponential	385	-0.33	0.95

Table 1: Fit coefficients to exponential and goodness of fit for the cardinality distribution in Fig. 2

We have seen that SAFARI is suited for computing multiple diagnoses of small probability. In the next section we will provide an alternative argument leading to similar conclusions.

5 DIAGNOSTIC METRICS

We next define the metrics used in the synthetic track of DXC'09. The two computational metrics M_{cpu} and M_{mem} are straightforward: M_{cpu} is the total amount of busy CPU time spent by SAFARI and M_{mem} is the peak amount of allocated memory (cf. (Kurtoglu *et al.*, 2009)). Building metrics that measure the "correctness" of a diagnosis is more involved and we discuss two of them: classification errors (M_{ia}) and utility (M_{utl}).

5.1 Classification Errors

M_{ia} is defined as follows:

$$M_{\text{ia}} = \sum_{c \in \text{COMPS}} \frac{\sum_{\omega \in \Omega} m_{\text{err}}(c, \omega, \omega^*)}{|\Omega| \cdot |\text{COMPS}|} \tag{10}$$

where $m_{\text{err}}(c, \omega, \omega^*)$ is defined as:

$$m_{\text{err}}(c, \omega, \omega^*) = \begin{cases} 0, & \text{if } \omega[c] = \omega^*[c] \\ 1, & \text{otherwise} \end{cases} \tag{11}$$

In (10) and (11) the injected fault is denoted as ω^* and $\omega[c]$ is the state of component c (healthy/faulty) in diagnosis ω.

Although M_{ia} has been applied successfully to the industrial track of DXC'09, it is considered as unsuitable for the synthetic track of DXC'09. The M_{ia} metric credits too much a diagnostic algorithm when, for example, this algorithm produces no diagnosis (in this case the diagnostic algorithm is assumed to have produced the "all healthy" diagnosis) and makes little intuitive sense with multiple diagnoses. Note that the DXC'09 industrial track contains fault scenarios of small cardinality (mostly single and double faults) and the system is sensor-rich, hence it is relatively easy to compute a single diagnostic candidate.

Instead of M_{ia} the DXC organizers have chosen to compute the utility metric M_{utl} for the synthetic track scenarios.

5.2 Utility Metric

Consider a truly injected fault $\omega^\star$ (a set of faulty components) and a diagnostic candidate ω. The number of truly faulty components that are improperly diagnosed by the diagnostic algorithm as healthy (false negatives) is $n = |\omega^\star \setminus \omega|$ (cf. Fig. 3). In general a diagnostician has to perform extra work to verify a diagnostic candidate ω, which must be reflected in the utility metric. We assume that he or she has access to a test oracle that states if a component c is healthy or faulty.

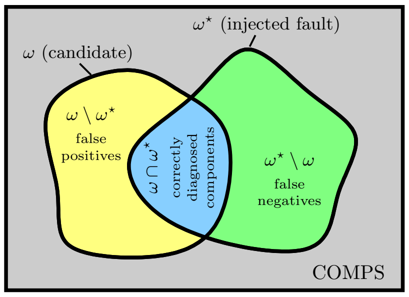
$\omega^\star$ (injected fault)
ω (candidate)
$\omega \setminus \omega^\star$
false positives
$\omega \cap \omega^\star$
correctly diagnosed components
$\omega^\star \setminus \omega$
false negatives
COMPS

Figure 3: Set operations on the diagnostic candidate ω and the injected fault $\omega^\star$

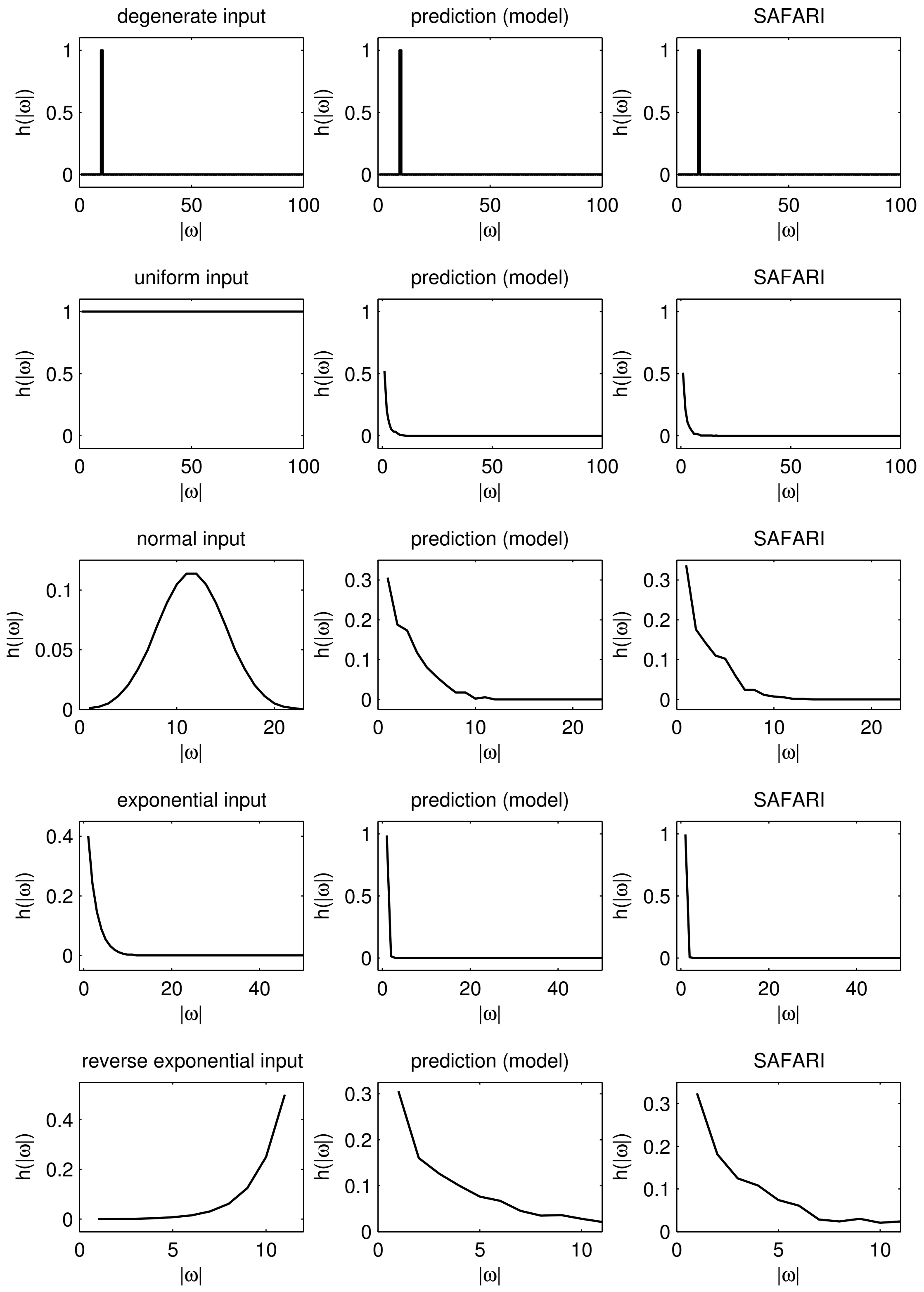

Figure 2: Predicted and actual cardinality distributions

We first determine what the expected number of tests a diagnostician has to perform to test all components in $\omega^\star \setminus \omega$ (the false negatives) if the diagnostician chooses untested components at random with uniform probability. In the worst case, the diagnostician has to test all the remaining $\text{COMPS} \setminus \omega$ components (the diagnostic algorithm has already determined the state of all components in ω). On average the situation is less worse. We denote $N = |\text{COMPS} \setminus \omega|$. N is the size of the "population" (of components to be tested). The inverse hypergeometric distribution

$$f(k,s,n,N) = \frac{\binom{n}{s-1}\binom{N-n}{x-s}(n-s+1)}{\binom{N}{x-1}(N-x+1)} \tag{12}$$

yields the probabilities for testing k healthy components before we find s faulty components out of the population (no repetitions). The expected value $\text{E}'[k]$ of $f(k,s,n,N)$ is:

$$\text{E}'[k] = \frac{s(N-n)}{n+1} \tag{13}$$

As we are interested in finding $s = n$ faulty components, the expected value $E'(n,N)$ becomes:

$$\text{E}'[k] = \frac{n(N-n)}{n+1} \tag{14}$$

The expected number of tests $\text{E}[t]$ (as opposed to the expected number of faulty components $\text{E}'[k]$) then becomes:

$$\text{E}[t] = \frac{n(N-n)+n}{n+1} = \frac{n(N+1)}{n+1} \tag{15}$$

Using $\text{E}[t]$ in a metric is not enough as it only captures the effort to "eliminate" (test) all false negatives. The size of the set of false positives is $\bar{n} = |\omega \setminus \omega^\star|$ (cf. Fig. 3). To find all false positives, the diagnostician has to test in the worst case all components in ω. Hence, the general population is $\bar{N} = |\omega|$. Repeating the argument for $\text{E}[t]$ we determine the expected number of tests for testing all false positives $\text{E}[\bar{t}]$.

Note that $0 \le \text{E}[t] \le N$ and $0 \le \text{E}[\bar{t}] \le \bar{N}$. Normalizing $\text{E}[t]$ and $\text{E}[\bar{t}]$ in the interval $[0; 0.5]$ and subtracting them from 1 (so bigger values means better) gives us the utility metric (per candidate):

$$m_{\text{utl}} = 1 - \frac{1}{2}\frac{\text{E}[t]}{N} - \frac{1}{2}\frac{\text{E}[\bar{t}]}{\bar{N}} = \tag{16}$$

$$= 1 - \frac{n(N-n)}{2N(n+1)} - \frac{\bar{n}(\bar{N}-\bar{n})}{2\bar{N}(\bar{n}+1)} \tag{17}$$

Table 2 summarizes all variables used in the formula of m_{utl}.

The utility metric (per scenario) is

$$M_{\text{utl}} = \sum_{\omega \in \Omega} W(\omega) m_{\text{utl}}(\omega^\star, \omega) \tag{18}$$

where $W(\omega)$ is the weight of a diagnosis ω such that

$$\sum_{\omega \in \Omega} W(\omega) = 1 \tag{19}$$

Var.	Set	Description
n	$\omega^\star \setminus \omega$	False positives.
N	$\text{COMPS} \setminus \omega$	The set of healthy components from the viewpoint of the diagnostic algorithm.
$\bar{n}$	$\omega \setminus \omega^\star$	False negatives.
$\bar{N}$	ω	The set of faulty components from the viewpoint of the diagnostic algorithm

Table 2: Notational summary of m_{utl}

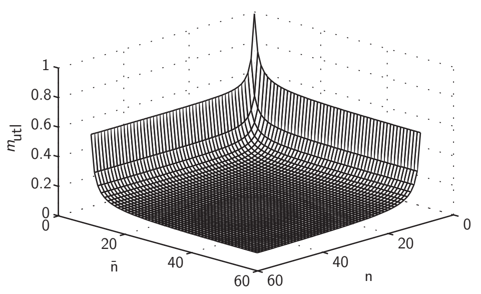

Figure 4: m_{utl} as a function of n and $\bar{n}$

All weights $W(\omega)$, $\omega \in \Omega$, are computed by the diagnostic algorithm.

Figure 4 plots m_{utl} for some ω and $\omega^\star$. Clearly, there is a global optimum $m_{\text{utl}} = 1$ for $n = 0$ and $\bar{n} = 0$, i.e., all components are classified correctly in ω.

5.3 Maximizing the Utility Metric

A diagnostic algorithm can maximize M_{utl} by computing a single diagnosis ω such that $\omega = \omega^\star$ (cf. Fig. 4). In the latter case, the weight of ω should be set to $W(\omega) = 1$. A diagnostic algorithm, however, has no way to know the truly injected diagnosis. Given a system description SD and an observation α, an uninformed diagnostic algorithm can at most compute the ambiguity group $\Omega(\text{SD} \wedge \alpha)$.

In the following conjecture we assume that $\omega^\star$ is a minimal-cardinality diagnosis, i.e., the faults in $\omega^\star$ do not mask.

Hypothesis 1. Given a system description SD, an observation α, and a truly injected fault $\omega^\star$, a diagnostic algorithm can maximize M_{utl} by computing $\Omega^{\le}(\text{SD} \wedge \alpha)$ and assigning equal weights to all diagnoses in $\Omega^{\le}(\text{SD} \wedge \alpha)$.

The intuition behind Hypothesis 1 is the following. An optimal diagnostic algorithm would compute an ambiguity group of minimal size, such that one of the diagnoses in the ambiguity group is the truly injected fault. It would then spread the weight amongst the diagnoses in this ambiguity group. A diagnosis of non-minimal-cardinality is clearly not the injected fault $\omega^\star$, hence adding it to the ambiguity group would be suboptimal.

In the previous sections we have shown that the ambiguity groups computed by SAFARI are highly likely to contain diagnoses of small cardinality, hence scoring high on M_{utl}. In what follows we will further support this claim with empirical results.

6 EXPERIMENTAL RESULTS

The DXC'09 synthetic track consists of the benchmark models of ISCAS85 circuits (Brglez and Fujiwara, 1985). These circuits are combinational, i.e., they contain no flip-flops or other memory elements. Note that the high-level structure of the ISCAS85 circuits, which can be beneficial to MBD analysis, has been flattened out. A reverse engineering effort had resulted in high-level Verilog models (Hansen *et al.*, 1999). Table 3 summarizes the circuits used in the synthetic DXC'09 track. In Table 3, $|\text{IN}|$ and $|\text{OUT}|$ denote the number of inputs and outputs respectively (at present SAFARI does not use input/output information for increasing the inference speed).

The size of the circuits in Table 3 can be reduced by using cones (Siddiqi and Huang, 2007) for computing single-component ambiguity groups (Kurtoglu *et al.*, 2009). To eliminate the need of expanding diagnoses containing faulty components inside cones, we have used the original (non-reduced) circuits.

Name	$\|\text{IN}\|$	$\|\text{OUT}\|$	$\|\text{COMPS}\|$	V	C
74182	9	5	19	47	75
74L85	11	3	33	77	118
74283	9	5	36	81	122
74181	14	8	65	144	228
c432	36	7	160	356	1 028
c499	41	32	202	445	1 428
c880	60	26	383	826	2 224
c1355	41	32	546	1 133	3 220
c1908	33	25	880	1 793	4 756
c2670	233	140	1 193	2 695	6 538
c3540	50	22	1 669	3 388	9 216
c5315	178	123	2 307	4 792	13 386
c6288	32	32	2 416	4 864	14 432
c7552	207	108	3 512	7 232	19 312

Table 3: ISCAS85 models (V and C denote the total number of variables and the number of clauses respectively)

We have configured SAFARI with $M = |\text{COMPS}|$ and $N = 10$. The value of N we have computed empirically. Increasing N to 20 or decreasing it showed no significant change in the metric results. We have also switched DPLL checking off, relying exclusively on BCP for the consistency checking of the candidate diagnoses. As a result of the BCP incompleteness, SAFARI produces inconsistent candidates, but this is rare (we have estimated less than 20% such candidates) and the overall effect on the metrics is positive.

6.1 Computing Multiple Minimal Diagnoses

We next show the results of some non-DXC'09 experiments supporting the claims made in Sec. 4. For that, we have first chosen 100 observations per circuit for which we could compute $|\Omega^{\leq}(\text{SD} \wedge \alpha)|$ with a deterministic algorithm like CDA* or HA* (mostly observations leading to single or double faults). We have then configured SAFARI with $M = |\text{COMPS}|$ and $N = 10|\Omega^{\leq}(\text{SD} \wedge \alpha)|$. Finally, from the diagnoses computed by SAFARI we have filtered the minimal-cardinality ones. The results are summarized in Table 4.

Name	$\|\Omega^{\leq}\|$	M_c	M_f
74182	$1-25$	100	0
74L85	$1-78$	99.2	2
74283	$1-48$	97.9	3
74181	$1-133$	97.4	1
c432	$1-99$	94.2	7.14
c499	$1-22$	78.5	1.51
c880	$2-646$	99.9	0
c1355	$5-2\,770$	79.4	1.02
c1908	$2-1\,447$	96.6	2.61
c2670	$1-76$	100	2.34
c3540	$1-384$	81.5	8.52
c5315	$1-235$	97.7	1.74
c6288	$1-154$	100	13.1
c7552	$1-490$	93.1	2.17

Table 4: % of all minimal-cardinality diagnoses computed by SAFARI

The data in Table 4 are to be interpreted as follows. The columns marked with $|\Omega^{\leq}|$ show the minimal and maximal number of minimal-cardinality diagnoses per model as computed by a deterministic algorithm. The columns M_c show the percentage of minimal-cardinality diagnoses returned by SAFARI (from all minimal-cardinality diagnoses) for those α for which $|\Omega^{\leq}(\text{SD} \wedge \alpha)| > 1$. The columns M_f show the percentage of observations for which SAFARI could not compute any minimal-cardinality diagnosis.

The results shown in Table 4 show that even for moderate values of N ($N \leq 27\,770$), SAFARI was capable of computing a significant portion of all minimal-cardinality diagnoses. This portion varies from 78.5% to 100% for weak-fault models and from 78% to 100% for strong-fault models. The percentage of cases in which SAFARI could not reach a minimal-cardinality diagnosis is limited (smaller than 13.55%) and is mainly in the cases in which there exists only one single-fault diagnosis. Note that even in the cases in which SAFARI cannot compute any minimal-cardinality diagnoses, the result of SAFARI can be useful. For example, a subset-minimal diagnosis of small cardinality differing in one or two literals only, still brings useful diagnostic information (a discussion on diagnostic metrics is beyond the scope of this paper).

6.2 Comparison to Other MBD Algorithms

Despite that M_{ia} is not appropriate for the synthetic track, we have computed it, and the results are shown in Table 5 (cf. discussion in Sec. 5). Note that M_{utl} is computed for each scenario. The "per system" metrics M_{UTL}, M_{CPU}, and M_{MEM} are M_{UTL}, M_{cpu}, and M_{mem} (respectively), averaged over all scenarios of a system.
Table 5 shows the results of all metrics for SAFARI as well as the number of scenarios for each circuit and the

		LYDIA			NGDE			RODON		
Name	|COMPS|	M_{CPU}	M_{MEM}	M_{UTL}	M_{CPU}	M_{MEM}	M_{UTL}	M_{CPU}	M_{MEM}	M_{UTL}
74182	19	51	154	0.418	6 335	11 540	0.4831	3 043	19 773	0.4153
74L85	33	68	223	0.3772	6 365	11 784	0.4454	3 888	20 979	0.3338
74283	36	60	229	0.2653	6 385	12 231	0.2665	5 351	20 637	0.2119
74181	65	64	401	0.2772	6 619	14 625	0.3337	12 527	25 432	0.2626
c432	160	115	878	0.2355	7 520	17 868	0.3789	22 621	36 811	0.2484
c499	202	130	1 094	0.1069	20 347	32 649	0.1248	23 504	39 872	0.0443
c880	383	203	1 945	0.0946	13 718	28 622	0.1016	20 347	43 687	0.0526
c1355	546	296	2 759	0.1059	22 550	37 930	0.0808	23 253	33 530	0.0470
c1908	880	538	4 134	0.0962	26 171	39 843	0.0709	27 718	38 557	0.0560
c2670	1 193	937	5 867	0.1860	20 537	61 722	0.2715	35 680	43 063	0.1281
c3540	1 669	1 674	7 900	0.1608	27 022	82 045	0.1346	–	–	–
c5315	2 307	3 091	11 316	0.0862	30 926	93 116	0.1443	–	–	–
c6288	2 416	3 530	12 037	0.2499	17 483	102 420	0.2916	–	–	–
c7552	3 512	11 817	16 679	0.1492	37 989	125 910	0.1523	–	–	–
Averaged	–	1 613	4 687	0.2006	17855	48 022	0.2343	12 709	23 024	0.0311

Table 6: DXC'09 results.

Name	M_{ia} total	# of scenarios	M_{ia} per scenario
74182	145.6	50	2.9
74L85	73.6	28	2.6
74283	107.3	30	3.6
74181	258.9	54	4.8
c432	253.4	45	5.6
c499	2 326.4	141	16.5
c880	3 897.8	198	19.7
c1355	1 551.7	98	15.8
c1908	2 302.8	127	18.1
c2670	2 419.8	168	14.4
c3540	373.3	36	10.4
c5315	7 658.6	248	30.9
c6288	2	1	2
c7552	3 103	176	17.6

Table 5: Isolation accuracy of SAFARI (metrics not used in DXC'09).

number of classification errors per scenario. It can be seen that M_{ia} depends on the number of diagnoses SAFARI produces and the latter depends on the parameter N in Alg. 1.

Table 6 shows a comparison of SAFARI to the two other DXC'09 synthetic track algorithms: NGDE (de Kleer, 2009) and RODON (Bunus *et al.*, 2009). It can be seen that SAFARI achieved better M_{CPU} and M_{MEM} than NGDE and RODON. M_{UTL} of SAFARI was worse than that of NGDE. It should be noted that RODON could not compute any results for the four largest ISCAS85 circuits due to a time limitation in DXC'09 (we believe that this can be easily overcome in subsequent competitions). Hence, the three algorithms (SAFARI, NGDE, and RODON) showed very similar results in the utility metrics.

7 CONCLUSION

In this paper we have discussed the properties of SAFARI in computing multiple minimal diagnoses. We have seen that SAFARI computes diagnoses of small cardinality with probability which is negatively exponential to the cardinality of the diagnoses. As a result, the set of diagnoses returned by SAFARI can be used instead the set of all minimal-cardinality diagnoses of a diagnosis system and an observation. Computing the set of all minimal-cardinality diagnoses is a policy which has been chosen by all synthetic track DXC'09 algorithms for maximizing the utility metric (the expected cost of repair).

To summarize the experimental results, SAFARI has achieved good performance on the isolation accuracy metrics while keeping the memory and CPU requirements very low. The low CPU and memory requirement is not a surprise considering the stochastic nature of SAFARI. Our results show that computing multiple-fault diagnoses close to the actually injected faults is practical with a cheap and simple stochastic algorithm.

As future work we would like to investigate the use of SAFARI for computing cardinality distributions while learning the termination criterion (cf. Sec. 4) instead of specifying the parameter N. This would give us a very efficient algorithm for counting the number of minimal-cardinality diagnoses, and (after extending the SAFARI framework to handle probabilities) estimating the entropy of the health space, the latter a key component in testing or probing algorithms.

REFERENCES

(Brglez and Fujiwara, 1985) Franc Brglez and Hideo Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. In *Proc. ISCAS'85*, pages 695–698, 1985.

(Bunus *et al.*, 2009) Peter Bunus, Olle Isaksson, Beate Frey, and Burkhard Münker. RODON - a model-based diagnosis approach for the DX diagnostic competition. In *Proc. DX'09*, pages 423–430, 2009.

(Davis *et al.*, 1962) Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

(de Kleer and Williams, 1987) Johan de Kleer and Brian Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

(de Kleer *et al.*, 1992) Johan de Kleer, Alan Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

(de Kleer, 1986) Johan de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28(2):197–224, 1986.

(de Kleer, 1990) Johan de Kleer. Using crude probability estimates to guide diagnosis. *Artificial Intelligence*, 45(3):381–291, 1990.

(de Kleer, 2009) Johan de Kleer. Minimum cardinality candidate generation. In *Proc. DX'09*, pages 397–402, 2009.

(Feldman and van Gemund, 2006) Alexander Feldman and Arjan van Gemund. A two-step hierarchical algorithm for model-based diagnosis. In *Proc. AAAI'06*, pages 827–833, July 2006.

(Feldman *et al.*, 2006) Alexander Feldman, Jurryt Pietersma, and Arjan van Gemund. All roads lead to fault diagnosis: Model-based reasoning with LYDIA. In *Proc. BNAIC'06*, October 2006.

(Feldman *et al.*, 2008a) Alexander Feldman, Gregory Provan, and Arjan van Gemund. Computing minimal diagnoses by greedy stochastic search. In *Proc. AAAI'08*, pages 911–918, 2008.

(Feldman *et al.*, 2008b) Alexander Feldman, Gregory Provan, and Arjan van Gemund. Computing observation vectors for max-fault min-cardinality diagnoses. In *Proc. AAAI'08*, July 2008.

(Forbus and de Kleer, 1993) Kenneth Forbus and Johan de Kleer. *Building Problem Solvers*. MIT Press, 1993.

(Hansen *et al.*, 1999) Mark Hansen, Hakan Yalcin, and John Hayes. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test*, 16(3):72–80, 1999.

(Kurtoglu *et al.*, 2009) Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, Arjan van Gemund, and Alexander Feldman. First international diagnosis competition - DXC'09. In *Proc. DX'09*, pages 383–396, 2009.

(McAllester, 1990) David McAllester. Truth maintenance. In *Proc. AAAI'90*, volume 2, pages 1109–1116, 1990.

(Siddiqi and Huang, 2007) Sajjad Siddiqi and Jinbo Huang. Hierarchical diagnosis of multiple faults. In *Proc. IJCAI'07*, pages 581–586, 2007.

(Williams and Ragno, 2007) Brian Williams and Robert Ragno. Conflict-directed A* and its role in model-based embedded systems. *Journal of Discrete Applied Mathematics*, 155(12):1562–1595, 2007.