Probabilistic Sequential Diagnosis by Compilation

Sajjad Siddiqi and **Jinbo Huang**
National ICT Australia and Australian National University
Canberra, ACT 0200 Australia
{sajjad.siddiqi, jinbo.huang}@nicta.com.au

Abstract

When a system behaves abnormally, a diagnosis is a set of system components whose failure explains the abnormality. It is known that compiling the system model into deterministic decomposable negation normal form (d-DNNF) allows efficient computation of the complete set of diagnoses. We extend this approach to *sequential diagnosis*, where a sequence of measurements is taken to narrow down the set of diagnoses until the actual faults are identified. We propose novel probabilistic heuristics to reduce the diagnostic cost, defined as the number of measurements. Our heuristics involve the posterior probabilities of component failures and the entropies of measurement points. The structure of the system is exploited so that a joint probability distribution over the faults and system variables is represented compactly as a Bayesian network, which is then compiled into d-DNNF. All posterior probabilities required are computed exactly and efficiently by traversing the d-DNNF. Finally, we scale the approach further by performing the diagnosis in a hierarchical fashion. Compared with the previous GDE framework, whose heuristic involves the entropy over the set of diagnoses and estimated posterior probabilities, we avoid the often impractical need to explicitly go through all diagnoses, and are able to compute the probabilities exactly. Experiments with ISCAS-85 circuits indicate that our approach can solve for the first time a set of multiple-fault diagnostic cases on large circuits, with good performance in terms of diagnostic cost.

1 Introduction

When a system behaves abnomormally, the task of *diagnosis* is to identify the reasons for the abnormality. For example, in the combinational circuit in Figure 1, given the inputs $P \wedge Q \wedge \neg R$, the output V should be 0, but it is actually 1 due to the faults at gates J and B.

Given a knowledge base modeling the behavior of a system comprising a set of components, along with the (abnormal) observed values of some system variables, *consistency-based diagnosis* computes a set of diagnoses, where a diagnosis is a set of components whose failure together with the observation is logically consistent with the system model. In our example, $\{V\}, \{K\}, \{A\}$, and $\{J, B\}$ are some of the diagnoses given the observation. In general, the number of diagnoses can be exponential in the number of system components, and only one of them will correspond to the set of actual faults.

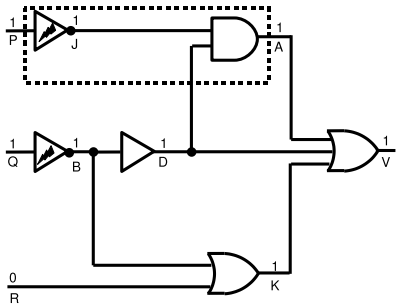


Figure 1: A faulty circuit.

In this paper, therefore, we consider the problem of *sequential diagnosis* (de Kleer & Williams 1987), where a sequence of measurements of system variables is taken to narrow down the set of diagnoses until the actual faults are identified. The goal is to reduce the diagnostic cost, defined as the number of measurements.

The heuristic proposed in the classical GDE (general diagnosis engine) framework (de Kleer & Williams 1987; de Kleer, Raiman, & Shirley 1992; de Kleer 2007) for this purpose required computing the Shannon's entropy of the probability distribution over the set of diagnoses, which was shown to be quite effective but can be infeasible to compute when the number of diagnoses is too large. In addition, the posterior probabilities of system variables with unknown values had to be estimated, due to the lack of an efficient method to compute them exactly.

We propose a new heuristic that does not require computing the entropy of diagnoses. Instead we consider the entropies of the system variables to be measured as well as the posterior probabilities of component failures. The idea is to select a component that has the highest posterior probability of failure (Heckerman, Breese, & Rommelse 1995) and from the variables of that component, measure the one that has the highest entropy.

To model the joint probability distribution over the system variables and component failures, we exploit the system structure and model each component as a conditional probability table and combine them into a *Bayesian network* (Pearl 1988), which is then compiled into deterministic decomposable negation normal form (d-DNNF) (Darwiche 2001; Darwiche & Marquis 2002). The d-DNNF size can remain

compact even when the number of diagnoses is large, thanks to the structure present in many systems. All the required posterior probabilities can be exactly computed by evaluating and differentiating the d-DNNF in time linear in the d-DNNF size (Darwiche 2003).

We scale our approach further to handle larger systems by using an idea similar to the abstraction based hierarchical diagnosis (Siddiqi & Huang 2007). Self-contained subsystems, called *cones*, are treated as single components and diagnosed only if they are found to be faulty in the top-level diagnosis. This significantly reduces the number of system components, allowing larger systems to be compiled and diagnosed. For example, the subcircuit in the dotted box in Figure 1 is a cone (with A as output and $\{P, D\}$ as inputs) which contains a fault. First, cone A, as a whole, is determined as faulty. It is only then that A is compiled separately and diagnosed.

Experiments on ISCAS-85 (Brglez & Fujiwara 1985) benchmark circuits indicate that we can solve, for the first time, a set of multiple-fault diagnostic cases on large circuits with good results, whereas the public version of GDE was unable to handle circuits with more than 19 gates.

2 Background and Previous Work

We start by formally modeling the system to be diagnosed by a joint probability distribution $Pr(\mathbf{X} \cup \mathbf{H})$ over a set of variables partitioned into $\mathbf{X}$ and $\mathbf{H}$.[1] Variables $\mathbf{X}$ are those whose values can be either observed or measured, and variables $\mathbf{H}$ are the *health* variables, one for each component describing its health mode (we shall also use these $\mathbf{H}$ variables to refer to components themselves). For example, for a system consisting of only the inverter J in Figure 1, we have $\mathbf{X} = \{P, J\}$ and $\mathbf{H} = \{okJ\}$. In what follows, we will assume for simplicity that all our variables are binary and that $H = 1$ indicates a healthy component for all $H \in \mathbf{H}$. Diagnosis starts in the initial (belief) state $I_0 = Pr(\mathbf{X} \cup \mathbf{H} \mid \mathbf{X_o} = \mathbf{x_o})$ where values $\mathbf{x_o}$ of some variables $\mathbf{X_o} \subseteq \mathbf{X}$ [2] are given by the observation, and we wish to reach a goal state $I_n = Pr(\mathbf{X} \cup \mathbf{H} \mid \mathbf{X_o} = \mathbf{x_o}, \mathbf{X_m} = \mathbf{x_m})$ after measuring the values $\mathbf{x_m}$ of some variables $\mathbf{X_m} \subseteq \mathbf{X} \backslash \mathbf{X_o}$, $|\mathbf{X_m}| = n$, one at a time, such that (the boldface $\mathbf{0}$ and $\mathbf{1}$ denote vectors of 0's and 1's):

$$\exists \mathbf{H_f} \subseteq \mathbf{H}, Pr(\mathbf{H_f} = \mathbf{0} \mid \mathbf{X_o} = \mathbf{x_o}, \mathbf{X_m} = \mathbf{x_m}) = 1 \text{ and}$$

$$Pr(\mathbf{H_f} = \mathbf{0}, \mathbf{H} \backslash \mathbf{H_f} = \mathbf{1} \mid \mathbf{X_o} = \mathbf{x_o}, \mathbf{X_m} = \mathbf{x_m}) > 0.$$

That is, in a goal state a set of components $\mathbf{H_f}$ are known to be faulty with certainty and no logical inconsistency arises if all other components are assumed to be healthy.[3] Two special casese are worth mentioning: (1) If the initial state I_0 satisfies the goal condition with $\mathbf{H_f} = \emptyset$ then the observation is normal and no diagnosis is required. (2) If the initial state I_0 satisfies the goal condition with some $\mathbf{H_f} \neq \emptyset$, then the observation is abnormal but the diagnosis is already completed (assuming that we are able to check probabilities as necessary), in other words, a sequence of length 0 solves the problem.

As in (de Kleer & Williams 1987), we assume that all measurements have unit cost. Hence the objective is to reach a goal state in the fewest measurements possible.

The classical GDE framework, on receiving an abnormal observation $\mathbf{X_o} = \mathbf{x_o}$, computes the set of all *minimal diagnoses*, which characterize the set of all diagnoses. It then computes the Shannon's entropy of the probability distribution over the set of diagnoses and proposes to measure a variable X whose value will reduce that entropy the most, on average. The idea is that the probability distribution over the diagnoses reflects the uncertainty over the actual faults, and the entropy captures the amount of this uncertainty. After a measurement is taken the entropy is updated by updating the posterior probabilities of the diagnoses, potentially reducing some of them to 0.

The results in (de Kleer, Raiman, & Shirley 1992) involving single-fault cases for ISCAS-85 circuits indicate that this method leads to measurement costs close to those of optimal policies. However, as described earlier, it has two major drawbacks. First, it can be impractical when the number of minimal diagnoses is large (when the fault cardinality is not restricted). Second, at each update step, the posterior probabilities of variables whose values are not predicted so far are only estimated due to the lack of an efficient method to compute them exactly.

3 Overview of New Method

We propose and implement a new method that addresses both drawbacks of GDE. From here on, we shall use combinational circuits as a concrete example of the type of systems we wish to diagnose. The method, however, applies as well to other types of systems as long as a probabilistic model is given that defines the behavior of the system.

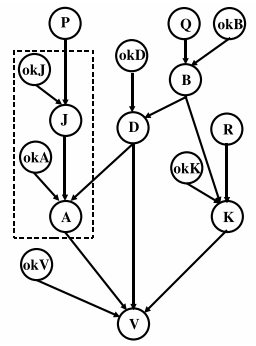
We model the circuit as a propositional formula in conjunctive normal form (CNF), and compile it, together with an abnormal observation, into d-DNNF, where we associate real numbers with the propositional variables, which represent the prior probabilities of gate failures given as part of the input to the diagnosis task. Because of the independence condition that exists among the gates, our d-DNNF compilation amounts to a compilation of a Bayesian network that compactly encodes the joint probability distribution over all the wires and faults of the circuit.

To select a measurement point, we use heuristics that do not require the enumeration of all diagnoses. Instead we only require the entropies of the wires together with the posterior probabilities of component failures. All probabilities required can be exactly computed simultaneously in a single traversal of the d-DNNF which performs partial differentiation (Darwiche 2003). Finally, we discuss the abstraction

[1] Variables are denoted by uppercase letters, their values by lowercase letters, vectors (or sets) of variables by boldface uppercase letters, and vectors of values by boldface lowercase letters.

[2] Note that we are abusing notation by using boldface uppercase letters to mean both sets and vectors.

[3] Other types of goal conditions are possible. For example, if the health states of all gates are to be determined with certainty, the condition will be that $Pr(H = 0 \mid \mathbf{X_o} = \mathbf{x_o}, \mathbf{X_m} = \mathbf{x_m})$ is 0 or 1 for all $H \in \mathbf{H}$. Such goals, though, are only possible to reach if strong fault models are given.



P	θ_P
1	0.5
0	0.5

okJ	θ_{okJ}
1	0.9
0	0.1

P	okJ	J	$\theta_{J\|P,okJ}$
1	1	1	0
1	1	0	1
1	0	1	0.5
1	0	0	0.5
0	1	1	1
0	1	0	0
0	0	1	0.5
0	0	0	0.5

Figure 2: Bayesian network for the circuit in Figure 1 and CPTs for P, J, and okJ.

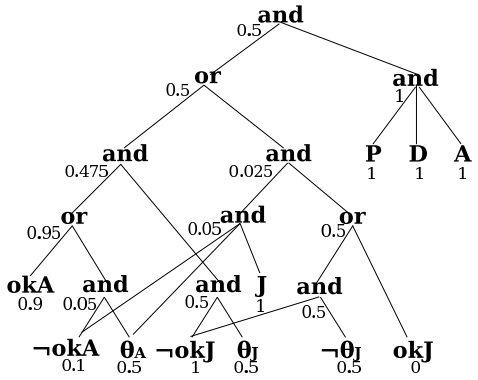


Figure 3: d-DNNF compilation of subcircuit (dotted) in Figure 1 given the observation $A \wedge P \wedge D$ and computation of the probability of evidence $\neg okJ$.

based hierarchical diagnosis which scales the approach further to handle larger circuits.

4 System Modeling and Compilation

In order to define a joint probability distribution $Pr(\mathbf{X} \cup \mathbf{H})$ over the circuit behavior, we first assume that the prior probability of failure $Pr(H = 0)$ is given for each gate $H \in \mathbf{H}$, as part of the input to the diagnosis task (de Kleer & Williams 1987). For example, the small table with two entries on the top-right of Figure 2 gives the prior probability of failure for gate J as 0.1. However, fault probabilities alone do not define the joint probability distribution $Pr(\mathbf{X} \cup \mathbf{H})$. In addition, we need to specify for each gate how its output is related to its inputs and health mode. A conditional probability table (CPT) for each gate does this job.

The CPT shown on the bottom-right of Figure 2, for example, defines the behavior of gate J: Each entry gives the probability of its output (J) being a particular value given the value of its input (P) and the value of its health variable (okJ). In case $okJ = 1$, the probabilities are always 0 or 1 as the behavior of a healthy gate is deterministic. The case of $okJ = 0$ defines the *fault model* of the gate, which is also part of the input to the diagnosis task. In our example, we assume that both output values have probability 0.5 when the gate is broken. Also, if a gate has more than two health modes, we can simply increase the number of health variables accordingly.

Given these tables, the joint probability distribution over the circuit behavior can be obtained by realizing that the gates of a circuit satisfy an independence property, known as the *Markov property*: Given its inputs and health mode, the output of a gate is independent of any wire which is not in the fan-out of the gate. This means that the circuit can be effectively treated as a Bayesian network in the straightforward way, by having a node for each wire and each health variable, and having an edge going from each input of a gate to its output, and also from the health variable of a gate to its output. Figure 2 (left) shows the result of this translation for the circuit in Figure 1.

The joint probability distribution encoded in the Bayesian network provides the basis for computing any posterior probabilities that we may need when proposing measurement points (by the *chain rule*). However, it does not provide an efficient way of doing so. Specifically, computing a posterior $Pr(X = x \mid \mathbf{Y} = \mathbf{y})$ given the values $\mathbf{y}$ of all the wires $\mathbf{Y}$ with known values involves summing out all variables other than X and $\mathbf{Y}$, which has a complexity exponential in the number of such variables if done naively.

4.1 Propositional Modeling

It is known that a Bayesian network can be encoded into CNF and then compiled into d-DNNF, which, if successful, allows posterior probabilities of all variables to be computed efficiently (Darwiche 2003). For the purposes of sequential diagnosis, we encode the Bayesian network as follows.

Consider the subcircuit in the dotted box in Figure 1 as an example, which can be modeled as the following five-clause CNF formula: $\neg okJ \vee \neg P \vee \neg J$, $\neg okJ \vee P \vee J$, $\neg okA \vee J \vee \neg A$, $\neg okA \vee D \vee \neg A$, $\neg okA \vee \neg J \vee \neg D \vee A$. Specifically, each signal of the circuit translates into a propositional variable (A, D, P, J), and for each gate, an extra variable is introduced to model its health (okA, okJ). The CNF formula is such that when all health variables are true, the remaining variables are constrained to model the functionality of the gates. For example, the first two clauses above are equivalent to $okJ \rightarrow (J \leftrightarrow \neg P)$, modeling the functionality of the inverter. In general, for each gate X, we have $okX \rightarrow$NORMALBEHAVIOR(X).

Note that the above formula fails to encode half of the CPT entries, where $okJ = 0$. In order to complete the encoding of the CPT of node J, we introduce an extra Boolean variable θ_J, and write $\neg okJ \rightarrow (J \leftrightarrow \theta_J)$.

Finally, the health variables (okA, okJ) are associated with the probabilities of the respective gates being healthy (0.9 in our experiments), and each θ-variable (θ_J) is associated with the probability of the corresponding gate giving an output of 1 when broken (0.5 in our experiments).

4.2 Compilation

Once we write the above clauses for all the gates, our CNF formula is ready for compilation into d-DNNF. d-DNNF is

a graph representation of a nested and/or expression where negation only appears at the leaves next to variables, children of every and-node have disjoint sets of variables (decomposability), and children of any or-node are mutually logically inconsistent (determinism). Figure 3 shows the d-DNNF compilation of the subcircuit in the dotted box of Figure 1 under the observation $A \wedge P \wedge D$.

After the compilation, the probability $Pr(\mathbf{E} = \mathbf{e})$ for an instantiation $\mathbf{e}$ of any set of variables $\mathbf{E}$ can be obtained by the following linear-time procedure: (i) Set all variables $\mathbf{E}$ to Boolean constants according to the instantiation $\mathbf{e}$, (ii) set all other literals to true except those that have numbers associated with them (negative literals are associated with 1 minus the corresponding numbers for the positive literals), and (iii) evaluate the d-DNNF by treating true as 1, false as 0, the remaining leaves as their associated numbers, or-nodes as additions, and and-nodes as multiplications. The number at the root will be $Pr(\mathbf{E} = \mathbf{e})$. For example, Figure 3 shows the computation of the probability of gate J being broken given the observation $A \wedge P \wedge D$. Furthermore, when the value of a variable becomes known (by measurement), the posterior probabilities of all variables can be updated simultaneously by differentiating the d-DNNF in time linear in the size of the d-DNNF (Darwiche 2003).

5 Probabilistic Sequential Diagnosis

Able to compute probabilities efficiently and exactly (following successful d-DNNF compilation), we are now ready to discuss the measurement point selection heuristic. We start with a definition of Shannon's entropy H, which is defined with respect to a probability distribution of a discrete random variable X ranging over values $x_1, x_2, \ldots, x_k$. Formally: $H(X) = -\sum_{i=1}^{k} Pr(X = x_i) \log Pr(X = x_i)$

Entropy measures the amount of uncertainty over the value of the random variable. It is maximal when all probabilities $Pr(X = x_i)$ are equal, and minimal when one of the probabilities is 1, corresponding nicely to our intuitive notion of the degree of uncertainty.

In GDE (de Kleer & Williams 1987; de Kleer 2007), the entropy is computed for the probability distribution over the set of diagnoses computed (i.e., the value of the random variable X here ranges over the set of all diagnoses). As discussed in those same papers, however, this entropy can be difficult to compute when the number of diagnoses is large.

5.1 Baseline Approach

We propose a new two-part heuristic that avoids this drawback. First, we consider the entropy of a candidate wire to be measured.

Heuristic Based on Entropy of Wire. Since a wire X only has two values, its entropy can be written as: $H(X) = -(p_x \log p_x + p_{\bar{x}} \log p_{\bar{x}})$, where $p_x = Pr(X = 1 \mid \mathbf{Y} = \mathbf{y})$ and $p_{\bar{x}} = Pr(X = 0 \mid \mathbf{Y} = \mathbf{y})$ are the posterior probabilities of X having values 1 and 0, respectively, given the values $\mathbf{y}$ of wires $\mathbf{Y}$ whose values are known.

While $H(X)$ captures the uncertainty over the value of the wire, we can also interpret it as the average amount of information gain provided by measuring the wire. Hence as a first idea we consider selecting a wire with maximal entropy for measurement at each step.

Improving Heuristic Accuracy. This idea alone, however, did not work very well in our initial experiments. As would be confirmed by subsequent experiments, this is large due to the fact that the space of all diagnoses is generally very large and can include a large number of unlikely diagnoses, which tends to compromise the accuracy of the estimate of information gain provided by the entropy.

The experiments to confirm this explanation are as follows. When the d-DNNF compilation is produced, and before it is used to compute probabilities, we prune the d-DNNF graph so that models (satisfying variable assignments) corresponding to diagnoses with more than k broken gates are removed.[4] The resulting d-DNNF can be pruned further as faults are identified, by subtracting the number of the identified faults from k.

We set the initial k to the number of actual faults in the experiments, and observed that a significant reduction of diagnostic cost resulted in almost all cases. This improved performance is apparently due to that the pruning updates the posterior probabilities of all variables, making them more accurate since many unlikely diagnoses have been eliminated.

In practice, however, the number of faults is not known beforehand and choosing an appropriate k for the pruning can be nontrivial (note that k need not be exactly the same as the number of actual faults for the pruning to help). Interestingly, the following heuristic appears to achieve a similar performance gain in an automatic way.

The final heuristic is as follows. We select a component that has the highest posterior probability of failure (an idea from Heckerman, Breese, & Rommelse 1995), and then from the variables of that component, measure the one that has the highest entropy. This heuristic does not require the above pruning of the d-DNNF, and appears to improve the diagnostic cost to a similar extent by focusing the measurement selection on the component most likely to be broken.

5.2 Hierarchical Approach

We now scale our approach further to handle larger systems using the idea of abstraction based hierarchical diagno-

[4] A complete pruning is not easy; however, an approximation can be achieved in time linear in the d-DNNF size, by a variant of the minimization procedure described in (Darwiche 2001). It requires associating two registers with each node n of the d-DNNF, namely, ur and dr, and a two-pass traversal through the d-DNNF described below.

Initialize $ur(n)$ to 0 and $dr(n)$ to -∞ (least possible value) for all n. Traverse the d-DNNF so that children are visited before parents and for every leaf node, set $ur(n)$ to 1 if n is a negated health variable and 0 otherwise; for every or node, set $ur(n)$ to the minimum of the values of ur of its children; for every and node set $ur(n)$ to the sum of the values of ur of its children.

Now traverse the d-DNNF so that parents are visited before children and set dr for the root node to the value k; for every or node, remove every child p of n for which $ur(p) > dr(n)$ and for every remaining child v set $dr(v)$ to $dr(n)$ if $dr(n) > dr(v)$; for every child p of every and node, let t_p be the sum of the values of ur of all the other children and set $dr(p)$ to the value t_p if $t_p > dr(p)$.

sis (Siddiqi & Huang 2007). The basic idea is that the compilation of the system model into d-DNNF will be more efficient and scalable when the number of system components is reduced. This can be achieved by abstraction, where subsystems, known as cones in the case of circuits, are treated as single components. An example of a cone is depicted in Figure 1, and for space constraints we refer the reader to (Siddiqi & Huang 2007) for a formal definition of cones. When cones are abstracted in this way, we only need a single health variable and failure probability for the entire cone, significantly reducing the size of the CNF encoding and the difficulty of compilation. Once a cone is identified as faulty in the top-level diagnosis, it can then be compiled and diagnosed, in a recursive fashion.

Siddiqi & Huang (2007), however, only deal with the task of computing all diagnoses, which does not involve probabilities or measurement selection. In our case, several additional intricacies are worth mentioning, particularly in the computation of prior failure probabilities for the cones and the way measurement points are selected, as outlined below.

Prior Failure Probabilities for Cones. When a cone is treated as a single gate, it is necessary to compute its prior probability of failure as a whole, given the prior probabilities of gates and cones inside it. The method we use is to compute the probability of the cone outputting a wrong value by creating two copies ϕ_1 and ϕ_2 of the cone, in CNF, where ϕ_1 models only the healthy behavior of the cone (without health variables), and ϕ_2 includes the faulty behavior as well (i.e., the full encoding described in Section 4.1). The outputs of both ϕ_1 and ϕ_2 are collected into an XOR-gate X. We then compute the probability $Pr(X = 1)$, which gives the probability of the outputs of ϕ_1 and ϕ_2 being different. We compute this probability by compiling the entire encoding into d-DNNF and evaluating it under $X = 1$.

Note that this procedure itself is also hierarchical, performed bottom-up with the probabilities for the inner cones computed before those for the cones containing them.

Measurement Point Selection. When diagnosing a cone, it is generally important to have full knowledge of the values of its inputs, before a final diagnosis is concluded. A diagnosis of a cone concluded with only partial knowledge of its inputs may never be part of any valid global diagnosis. The reason is that the diagnosis of the cone assumes that the unknown inputs can take either value, while in reality their values may become fixed when wires in other parts of the circuit are measured.

To avoid this situation while retaining the effectiveness of the heuristic, we modify the measurement point selection as follows when diagnosing a cone. After selecting a gate with the highest probability of failure, we consider the wires of that gate *plus* the inputs of the cone, and measure the one with the highest entropy. We do not conclude a diagnosis for the cone until values of all its inputs become known (through measurement or deduction), except when the health of all the gates in the cone has been determined without knowing all the inputs to the cone (it is possible to identify a faulty gate, and with strong fault models also a healthy gate, without knowing all its inputs).

We omit a formal description of the detailed hierarchical sequential diagnosis algorithm for space constraints, and proceed now to describe a thorough empirical evaluation of the new sequential diagnosis algorithms.

size	**system**	**single- fault**		**double-fault**		**triple-fault**	
		cost	*time*	*cost*	*time*	*cost*	*time*
13	GDE	3.6	2.0	3.8	1.81	4.0	1.9
	SDC	3.7	0.01	3.3	0.01	2.8	0.01
14	GDE	3.5	6.66	3.3	15.1	3.0	14
	SDC	4.0	0.01	3.2	0.01	2.7	0.01
15	GDE	3.4	111	3.5	88	4.3	299
	SDC	4.0	0.01	3.4	0.01	3.7	0.01
16	GDE	3.3	398	3.5	556	3.2	509
	SDC	3.6	0.01	3.4	0.01	2.8	0.01
17	GDE	3.7	2876	4.6	4103	4.5	2067
	SDC	4.0	0.01	4.0	0.01	4.0	0.01

Table 1: Comparison with GDE.

6 Experimental Results

We have implemented a new diagnostic system, which we refer to as SDC, that supports both the baseline and the hierarchical approach described in Section 5. The d-DNNF compilation is done using a publicly available d-DNNF compiler (Darwiche 2004; 2005). All experiments were conducted, using ISCAS-85 benchmark circuits, on a cluster of 32 computers consisting of two types of (comparable) CPUs, Intel Core Duo 2.4 GHz and AMD Athlon 64 X2 Dual Core Processor 4600+, both with 4 GB of RAM running Linux. A time limit of 24 hours was imposed on each test case.

The diagnostic system is given a faulty circuit where some gates give incorrect outputs. The inputs and outputs of the circuit are observed, and actual values of its wires are used to simulate the results of taking measurements. We use $Pr(okX = 1) = 0.9$ for all gates X of the circuit. Note that such cases, where all gates fail with equal probability, are conceivably harder to solve as the diagnoses will tend to be less differentiable. Then, for each gate, the two output values are given equal probability when the gate is faulty. Again, this will tend to make the cases harder to solve due to the high degree of uncertainty.

For each circuit and fault cardinality, we report the cost (number of measurements taken) and time (in seconds) to locate the faults, averaged over all test cases used.

6.1 Comparison with GDE

We used the publicly available version of GDE (Forbus & de Kleer 1993) for the comparison, downloadable from http://www.qrg.northwestern.edu/BPS/readme.html.

It uses ATCON, a constraint language developed using the LISP programming language, to represent diagnostic problem cases. A detailed account of this language is available in (Forbus & de Kleer 1993). Further, it employs an interactive user interface that proposes measurement points with their respective costs and lets the user enter outcomes of measurements.

For the purpose of comparison we translated our problem descriptions to the language accepted by GDE, and also

circuit	system	min	single-fault		double-fault		five-fault	
			cost	*time*	*cost*	*time*	*cost*	*time*
c432	RAND	*no*	92.5	16.0	93.3	16.2	101.8	22.4
		yes	4.6	11.3	34.5	11.7	86.5	12.8
(160 gates)	SDC	*no*	6.7	11.7	6.3	12.2	8.8	12.8
		yes	4.3	11	5.0	11.5	8.7	11.8
c499	RAND	*no*	110.0	0.6	106.5	0.6	124.0	0.7
		yes	5.4	0.2	18.8	0.2	87.5	0.6
(202 gates)	SDC	*no*	6.5	0.2	4.4	0.2	6.8	0.2
		yes	4.7	0.2	3.2	0.2	6.4	0.3
c880	RAND	*no*	220.3	1.5	232.5	1.4	268.2	1.5
		yes	5.4	0.2	43.8	0.3	185.2	1.0
(383 gates)	SDC	*no*	10.5	0.2	9.3	0.3	15.5	0.4
		yes	5.5	0.2	6.6	0.2	13.8	0.4
c1355	RAND	*no*	328.5	3.6	334.5	3.3	371.7	3.6
		yes	7.4	0.4	51.3	1.0	278.0	2.5
(546 gates)	SDC	*no*	33.8	1.0	15.0	0.6	16.7	0.6
		yes	8.0	0.5	10.0	0.5	15.5	0.5
c1908	RAND	*no*	448.0	19395	780.0	17137	689.0	30134
		yes	8.6	14052	16.7	13686	3.0	6613
(880 gates)	SDC	*no*	63.6	14375	70.0	13281	61.2	24971
		yes	7.8	10954	17.5	17533	60.2	42974

Table 2: Results on larger benchmarks.

modified GDE to automatically read in the measurement outcomes from the input problem description. We also compiled the LISP code to machine dependent binary code using the native C compiler to improve run-time performance.

GDE quickly ran out of memory even on the smalles circuit (c432) in ISCAS-85. We observed that the *Assumption Based Truth Maintenance System* (Forbus & de Kleer 1993), used by GDE as a reasoning system to generate minimal diagnoses, actually generated exponentially many sets of assumptions (about values of system variables, called environments) before running out of memory.[5]

To enable a useful comparison, we extracted a set of small subcircuits from the ISCAS-85 circuits: 50 circuits of size 13, 14, 15 and 16, and 10 circuits of size 17. For each circuit we randomly generated 5 single-fault, 5 double-fault and 5 triple-fault scenarios, and one test case (input/output vector) for each fault scenario.

Table 1 summarizes the comparison between GDE and SDC on these benchmarks, where the first column shows the size the circuit. It is clear that the running time of GDE increases by roughly an order of magnitude with an increase of just one gate in the circuit size, manifesting the exponential complexity of the algorithm. SDC performs as well as GDE in terms of diagnostic cost, and solves every case instantly while GDE takes up to more than an hour.

6.2 Larger Benchmarks

To evaluate the performance of SDC on the larger ISCAS-85 circuits, we have again conducted three sets of experiments, this time involving single, double, and five faults, respectively. For single faults, we simulated the equal prior probability of faults by generating n fault scenarios for each circuit, where n equals the number of gates in the circuit: Each scenario contains a different faulty gate. We then randomly generated 25 test cases for each of these n scenarios. Doing the same for double- and five-fault scenarios would not be practical due to the large number of combinations, so for each circuit we simply generated 1000 random scenarios with the given fault cardinality and a random test case for each scenario. For the largest circuits, c1908 and c2670, we only generated 10 fault scenarios under each fault cardinality and one test case for each fault scenario.

As the version of GDE available to us is unable to handle these circuits, in order to provide a systematic reference point for comparison we have implemented a random strategy where a random order of measurement points is generated for each circuit and used for all the test cases. This strategy also uses the d-DNNF to check whether the stopping criteria have been met.

Table 2 shows the comparison between the random strategy and SDC (using the baseline approach). For both systems we ran the same set of experiments with and without pruning the d-DNNF (using the known fault cardinality as described in Section 5.1). The third column ("min", for "minimization") of table indicates whether this was done. For c1908, out of the 10 test cases, only 5 could be solved for single-fault, 4 for double-fault, and 4 for five-fault. None of the test cases for c2670 could be solved. All failures occurred during the compilation phase, and hence affected both the random strategy and SDC.

It is clear from Table 2 that the diagnostic cost is significantly lower with SDC than with the random strategy whether or not pruning has been used. It is also interesting to note that pruning significantly reduces the diagnostic cost for the random strategy, but has much less effect with SDC. As discussed earlier (and confirmed by a separate set of experiments which we omit), the combination of failure probabilities and wire entropies appears to achieve a similar effect to pruning. There are several cases (c1355 single-fault, c1908 single-fault, and c1908 double-fault), however, where the pruning provides substantial further improvement for SDC. Finally, one surprising exception is seen in the case of c1908 five-fault, where the random strategy with pruning did remarkably well and SDC did remarkably poorly. While the former is likely due to luck (only 10 cases were used here), the latter suggests room for further improvement with the heuristic of SDC.

6.3 Hierarchical Method

We now report, in Table 3, the results of repeating the same experiments with SDC using the hierarchical approach.

Most notably, the running time significantly reduces for c1908, the largest that could be handled by SDC using the baseline approach, and we are now able to handle c2670 (solving 27 of the 30 cases between the three categories) as well as 11 more cases for c1908.

In terms of diagnostic cost, it can be seen that in most cases the hierarchical approach is either comparable to or better than the baseline approach, except in the case of c432.

[5] Results in (de Kleer, Raiman, & Shirley 1992) were obtained by using a simple reasoning system which assumes a single fault, in which case the number of diagnoses is bounded by the number of gates.

circuit	min	single- fault		double-fault		five-fault	
		cost	*time*	*cost*	*time*	*cost*	*time*
c432	*no*	15.3	0.4	15.3	0.4	20.3	0.6
(64 cones)	*yes*	4.5	0.3	9.8	0.3	19.8	0.4
c499	*no*	7.0	0.2	5.6	0.1	9.1	0.2
(90 cones)	*yes*	4.6	0.1	3.9	0.1	8.3	0.1
c880	*no*	9.1	0.1	9.9	0.2	16.4	0.2
(177 cones)	*yes*	5.6	0.1	7.6	0.1	15.5	0.2
c1355	*no*	8.0	0.2	7.5	0.2	11.4	0.3
(162 cones)	*yes*	5.8	0.2	6.0	0.2	11.0	0.2
c1908	*no*	20.6	400	25.6	542	33.0	883
(374 cones)	*yes*	6.2	339	17.5	439	33.6	572
c2670	*no*	18.2	514	17.8	299	22.7	199
(580 cones)	*yes*	10.4	426	8.8	201	22.2	149

Table 3: Results of hierarchical approach.

Note that pruning helps further reduce the diagnostic cost to various degrees as with the baseline approach. As mentioned earlier, in practice choosing the appropriate k for pruning may be nontrivial, suggesting an interesting topic for future work.

7 Conclusion and Related and Future Work

We have presented a new system, called SDC, for sequential diagnosis that significantly advances the state of the art by solving, for the first time, a set of nontrivial multiple-fault diagnostic cases on large benchmark circuits. On the small benchmarks that can be solved by the previous GDE system, SDC also exhibits much higher efficiency and comparable performance in terms of diagnostic cost.

A recent line of related work is (Flesch, Lucas, & van der Weide 2007), where the authors propose a new framework to integrate probabilistic reasoning into model-based diagnosis. The framework is based upon the notion of *conflict measure*, which originated as a tool for the detection of conflicts between an observation and a given Bayesian network (Jensen 2001). When a system is modeled as a Bayesian network for diagnostic reasoning, it is possible to use this conflict measure to differentiate between diagnoses according to their degree of consistency with a given set of observations. This work, however, does not address the problem of locating actual faults by taking measurements.

Based on the GDE framework, de Kleer (2007) studies the sensitivity of diagnostic cost to what is called the ϵ*-policy*, which is the policy that quantifies how the posterior probabilities of diagnoses are to be estimated when GDE computes its heuristic. In our case, probabilities of diagnoses are not required at all, and the other probabilities that are required can all be computed exactly by evaluating and differentiating the d-DNNF. Nevertheless, our algorithm can be sensitive to the initial probabilistic model given and sensitivity analysis in this regard may lead to interesting findings.

Other topics for future work include extensions to cases where measurements have varying costs, and the feasibility of finding optimal measurement selection policies.

Acknowledgments

Thanks to Brian Williams for help with obtaining a copy of GDE and to the anonymous reviewers for commenting on an earlier version of this paper. National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

Brglez, F., and Fujiwara, H. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 695–698.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17:229–264.

Darwiche, A. 2001. Decomposable negation normal form. *Journal of the ACM* 48(4):608–647.

Darwiche, A. 2003. A differential approach to inference in bayesian networks. *Journal of the ACM* 50(3):280–305.

Darwiche, A. 2004. New advances in compiling CNF into decomposable negation normal form. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, 328–332.

Darwiche, A. 2005. The C2D compiler user manual. Technical Report D-147, Computer Science Department, UCLA. http://reasoning.cs.ucla.edu/c2d/.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Journal of Artificial Intelligence* 32(1):97–130.

de Kleer, J.; Raiman, O.; and Shirley, M. 1992. One step lookahead is pretty good. In *Readings in model-based diagnosis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 138–142.

de Kleer, J. 2007. Improved analysis of probability estimates in model-based diagnosis. In *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

Flesch, I.; Lucas, P.; and van der Weide, T. 2007. Conflict-based diagnosis: Adding uncertainty to model-based diagnosis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 380–385.

Forbus, K. D., and de Kleer, J. 1993. *Building problem solvers*. Cambridge, MA, USA: MIT Press.

Heckerman, D.; Breese, J. S.; and Rommelse, K. 1995. Decision-theoretic troubleshooting. *Communications of the ACM* 38(3):49–57.

Jensen, F. V. 2001. *Bayesian Networks and Decision Graphs*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Siddiqi, S., and Huang, J. 2007. Hierarchical diagnosis of multiple faults. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 581–586.