New Advances in Sequential Diagnosis

Sajjad Siddiqi
National University of Sciences and Technology
Islamabad, Pakistan
sajjad.ahmed@seecs.edu.pk

Jinbo Huang
NICTA and Australian National University
Canberra, Australia
jinbo.huang@nicta.com.au

Abstract

Sequential diagnosis takes measurements of an abnormal system to identify faulty components, where the goal is to reduce the *diagnostic cost*, defined here as the number of measurements. To propose measurement points, previous work employs a heuristic based on reducing the entropy over a set of *diagnoses*, which can be impractical when the set of diagnoses is too large. Focusing on a smaller set of *probable* diagnoses scales the approach but generally leads to increased diagnostic cost. We propose a new diagnostic framework employing three new techniques—a more efficient heuristic for measurement point selection, abstraction-based sequential diagnosis, and component cloning—which scales to large systems with good performance in terms of diagnostic cost.

Introduction

Consider the combinational circuit in Figure 1. Given the inputs $P \wedge Q \wedge \neg R$, the output V should be 0, but is actually 1 due to the faults at gates J and B. Some of the diagnoses under this observation are: $\{V\}, \{K\}, \{A\}$, and $\{J, B\}$, and only one of them corresponds to the set of actual faults.

In *sequential diagnosis*, measurements of system variables are taken to isolate the actual faults (de Kleer and Williams 1987). To propose measurement points, the GDE (general diagnosis engine) framework (de Kleer and Williams 1987; de Kleer 1992; de Kleer, Raiman, and Shirley 1992; de Kleer 2006) considers a heuristic based on reducing the entropy over a set of computed diagnoses which is either the set of *minimum-cardinality diagnoses* or a set of *probable/leading diagnoses*. This approach can be impractical when the set of diagnoses is too large. Focusing on a smaller set of probable diagnoses scales the approach but can increase the likelihood of irrelevant measurements and generally leads to increased average diagnostic costs (de Kleer 1992).

We propose a new diagnostic framework employing three new techniques, which scales to much larger systems with good performance in terms of diagnostic cost. First, we propose a new heuristic that considers the entropies of the system variables to be measured as well as the posterior probabilities of component failures. To compute probabilities, the system is modeled as a *Bayesian network* (Pearl 1988) and compiled into *deterministic decomposable negation normal form* (d-DNNF) (Darwiche 2001; Darwiche and Marquis 2002; Darwiche 2003).

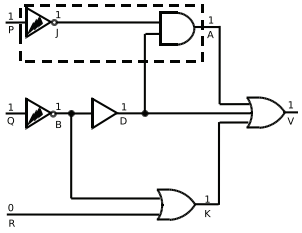

Figure 1: A faulty circuit.

Second, we extend *hierarchical diagnosis*, a technique based upon system abstraction (Siddiqi and Huang 2007), to handle probabilities so that it can be applied to sequential diagnosis to allow larger systems to be diagnosed.

Finally, when the abstraction of a system is still too large to be compiled, we propose a new method of *component cloning* that converts the system into one that has a smaller abstraction.

We use combinational circuits as an example of the type of systems we wish to diagnose. Our approach, 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 now present the new techniques we have introduced starting with the system modeling and compilation method that underlies our new diangostic system.

System Modeling and Compilation

Suppose that the system to be diagnosed is formally modeled by a joint probability distribution $Pr(\mathbf{X} \cup \mathbf{H})$ over a set of variables partitioned into $\mathbf{X}$ and $\mathbf{H}$. 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.

The modeling of systems as Bayesian networks has been discussed in (Siddiqi and Huang 2008). Here we only describe the encoding of Bayesian networks we use, which is analogous to what is described in (Darwiche 2003).

Consider the subcircuit in the dotted box in Figure 1 as an example, which can be modeled as the following formula:

$okJ \rightarrow (J \leftrightarrow \neg P)$, $okA \rightarrow (A \leftrightarrow (J \wedge D))$. 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 formula is such that when all health variables are true, the remaining variables are constrained to model the functionality of the gates. In general, for each component X, we have $okX \rightarrow$ NORMALBEHAVIOR(X). In addition, 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).

Once all components are encoded as described above, the union (conjunction) of the formulas is compiled into d-DNNF. The required probabilities can be exactly computed by evaluating and differentiating the d-DNNF in time linear in its size (Darwiche 2003).

We now present our baseline diagnosis approach and propose a new measurement selection heuristic.

New Measurement Point Selection

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 $\mathbf{X_o} \subseteq \mathbf{X}$ (we are using boldface uppercase letters to mean both sets and vectors) 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): $\exists \mathbf{H_f} \subseteq \mathbf{H}, Pr(\mathbf{H_f} = \mathbf{0} \mid \mathbf{X_o} = \mathbf{x_o}, \mathbf{X_m} = \mathbf{x_m}) = 1$ 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.

Two special cases 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.

An optimal solution to sequential diagnosis would be a *policy* (Heckerman, Breese, and Rommelse 1995), which is intractable to compute in general. Therefore, we follow the approach of heuristic measurement point selection based on Shannon's entropy as in previous work. We consider the entropy of a candidate variable to be measured, which provides the average amount of information gain provided by measuring the variable. Therefore we first consider measuring a variable with maximal entropy at each step.

This idea alone, however, did not work very well in our initial experiments. This is largely due to the fact that the (implicit) 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 information gain provided by the entropy. The experiments to confirm this explanation are as follows.

Before the d-DNNF is used to compute probabilities, we prune it so that models (satisfying variable assignments) corresponding to diagnoses with more than k broken components are removed.[1] 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 is apparently due to the fact may unlikely diagnoses get eliminated making the posterior probabilities of variables more accurate.

However, 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, which is the one we actually use, can achieve a similar performance gain in an automatic way: We select a component that has the highest posterior probability of failure (an idea from (Heckerman, Breese, and Rommelse 1995); discussed later), 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.

Algorithm 1 Sequential diagnosis

function SD($\mathbf{C}$, Δ, $\mathbf{D}$, $\mathbf{y}$, k)
inputs: {$\mathbf{C}$: system}, {Δ: d-DNNF}, {$\mathbf{y}$: measurements}, {k: fault cardinality}, {$\mathbf{D}$: ordered set of known faults}
output: {pair$< \mathbf{D}, \mathbf{y} >$}

1: REDUCE (Δ, $\mathbf{D}$, $k - |\mathbf{D}|$) if $\mathbf{D}$ has changed
2: Given $\mathbf{y}$ on variables $\mathbf{Y}$, EVALUATE (Δ, $\mathbf{y}$) to obtain $Pr(\mathbf{y})$
3: DIFFERENTIATE (Δ) to obtain $Pr(X = 1, \mathbf{y})$ $\forall$ variables X
4: Deduce fault as $\mathbf{D} = \mathbf{D} \cup \{X : Pr(okX = 1, \mathbf{y}) = 0\}$
5: **if** $\mathbf{D}$ has changed && MEETSCRITERIA(Δ,$\mathbf{D}$,$\mathbf{y}$) **then**
6: **return** $< \mathbf{D}, \mathbf{y} >$
7: Measure variable X which is the best under a given heuristic
8: Add the measured value x of X to $\mathbf{y}$, and go back to line 1

The Algorithm We start by encoding the system as a logical formula as discussed earlier, where a subset of the variables are associated with numbers representing the prior fault probabilities and probabilities involved in the fault models of the components, which is then compiled into d-DNNF Δ.

The overall sequential diagnosis process we propose is summarized in Algorithm 1. The inputs are a system $\mathbf{C}$, its d-DNNF compilation Δ, the set of faults $\mathbf{D}$ (which is empty but will be used in the hierarchical approach), a set of known values $\mathbf{y}$ of variables, and an integer k specifying the fault cardinality bound (this is for running the model pruning experiments described earlier, and is not required for diagnosis using our final heuristic). We reduce Δ by pruning some models (line 1) when the fault cardinality bound k is given. We then evaluate (line 2) and differentiate (line 3) Δ, select

[1] 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 described in (Darwiche 2001).

a measurement point and take the measurement (line 7), and repeat the process (line 8) until the stopping criteria are met (line 5).

The stopping criteria on Line 5 are given earlier as the goal condition, i.e., we stop when the abnormal observation is explained by all the faulty components $\mathbf{D}$ already identified assuming that other components are healthy. A faulty component X is identified when $Pr(okX = 1, \mathbf{y}) = 0$ where $\mathbf{y}$ are the values of variables that are already known, and as mentioned earlier these probabilities are obtained for all variables simultaneously in the d-DNNF differentiation process. Finally, the condition that the current set of faulty components, with health modes $\mathbf{H_f}$, explains the observation is satisfied when $Pr(\mathbf{H_f} = \mathbf{0}, \mathbf{H} \backslash \mathbf{H_f} = \mathbf{1}, \mathbf{y}) > 0$, which is checked by a single evaluation of the original d-DNNF. The algorithm returns the actual faults together with the new set of known values of variables (line 6).

Abstraction

We now scale our approach to handle larger systems using the idea of abstraction based hierarchical diagnosis (Siddiqi and 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, are treated as single components. The objective here is to use a single health variable and failure probability for the entire cone, hence significantly reducing the size of the 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. 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.

In (Siddiqi and Huang 2007), we only dealt with computing minimum-cardinality diagnoses, which does not involve probabilities or measurement selection. In the context of sequential diagnosis, several additional techniques have been introduced, particularly in the computation of prior failure probabilities for the cones and the way measurement points are selected, outlined below.

Propositional Encoding We start by discussing the hierarchical encoding for probabilistic reasoning, which is similar to the hierarchical encoding presented in (Siddiqi and Huang 2007). Specifically, for the diagnosis of the abstraction $\mathbf{A_C}$ of the given system $\mathbf{C}$, only the components in $\mathbf{A_C} \backslash \mathbf{I_C}$ will have extra health variables, which are the gates $\{A, B, D, K, V\}$ in our example ($\mathbf{I_C}$ stands for the set of inputs of the system $\mathbf{C}$).

In addition, we define the failure of a cone to be when it outputs the wrong value, and introduce extra clauses to model the abnormal behavior of the cone. For example, the encoding given earlier for cone A in Figure 1 (in the dotted box) is as follows: $J \leftrightarrow \neg P$, $okA \rightarrow (A \leftrightarrow (J \wedge D))$, $\neg okA \rightarrow (A \not\leftrightarrow (J \wedge D))$.

The first part of the formula encodes the normal behavior of gate J (without a health variable); the next encodes the normal behavior of the cone; the last encodes that the cone outputs a wrong value when it fails. Other gates (that are not roots of cones) in the abstraction $\mathbf{A_C}$ are encoded normally.

Note that the formulas for all the components in a cone together provide the conditional probability of the cone's output given the health and inputs of the cone, instead of the health and inputs of the component at the root of the cone. For example, the above encoding is meant to provide the conditional probability of A given P, D, and okA (instead of J, D, and okA), where okA represents the health mode of the whole cone and is associated with its prior failure probability, which is initially unknown to us and has to be computed for all cones (explained below). Such an encoding of the whole system provides a joint probability distribution over the variables $\mathbf{A_C} \cup \mathbf{I_C} \cup \mathbf{H}$, where $\mathbf{H} = \{okX \mid X \in \mathbf{A_C} \backslash \mathbf{I_C}\}$.

Prior Failure Probabilities for Cones When a cone is treated as a single component, its prior probability of failure as a whole can be computed given the prior probabilities of components and cones inside it. We do this by creating two copies Δ_h and Δ_f of the cone, where Δ_h models only the healthy behavior of the cone (without health variables), and Δ_f includes the faulty behavior as well (i.e., the full encoding described earlier). The outputs of both Δ_h and Δ_f are collected into an XOR-gate X(when the output of XOR-gate X equals 1, both of its inputs are forced to be different in value). We then compute the probability $Pr(X = 1)$ giving the probability of the outputs of Δ_h and Δ_f being different. The probability is computed by compiling this encoding into d-DNNF and evaluating it under $X = 1$.

Note that this procedure itself is also abstraction based and hierarchical, performed bottom-up with the probabilities for the inner cones computed before those for the outer ones. Also note that it is performed only once per system as a preprocessing step.

Measurement Point Selection and Stopping Criteria In principle, the measurement selection and the stopping criteria are the same as in the baseline method; however, a couple of details are worth mentioning.

First, when diagnosing the abstraction of a given system (or cone) $\mathbf{C}$, the measurement candidates are restricted to variables $\mathbf{A_C} \cup \mathbf{I_C}$, ignoring the internal variables of the maximal cones—those are only measured if a cone as a whole has been found faulty.

Second, it is generally important to have full knowledge of the values of cone's inputs before a final diagnosis of the cone is concluded. A diagnosis of a cone concluded with only partial knowledge of its inputs may exclude some faults that are vital to the validity of 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 be fixed when variables in other parts of the system are measured, causing the diagnosis of certain cones to become invalid, and possibly requiring the affected cones to be diagnosed once again to meet the global stopping criteria.

To avoid this situation while retaining the effectiveness of the heuristic, we modify the measurement point selection as

Algorithm 2 Hierarchical sequential diagnosis

function HSD($\mathbf{C}$, $\mathbf{u_C}$, k)
inputs: {$\mathbf{C}$: system},{$\mathbf{u_C}$: obs. across system} {k: fault cardinality}
local variables: {$\mathbf{B}, \mathbf{D}, \mathbf{T}$: set of components} {$\mathbf{y}, \mathbf{z}, \mathbf{u_G}$: set of measurements} {i, k' : integer}
output: {pair$< \mathbf{D}, \mathbf{u_C} >$}

1: $\Delta \leftarrow$ COMPILE2DDNNF ($\mathbf{A_C}$, $\mathbf{u_C}$)
2: $i \leftarrow 0$, $\mathbf{D} \leftarrow \phi$, $\mathbf{y} \leftarrow \mathbf{u_C}$
3: $< \mathbf{B}, \mathbf{y} > \leftarrow$ SD ($\mathbf{C}$, Δ, $\mathbf{B}$, $\mathbf{y}$, k)
4: **for** $\{; i < |\mathbf{B}|; i++\}$ **do**
5: $G \leftarrow$ELEMENT ($\mathbf{B}$, i)
6: **if** G is a cone **then**
7: $\mathbf{z} \leftarrow \mathbf{y} \cup$ IMPLICATIONS (Δ, $\mathbf{y}$)
8: $\mathbf{u_G} \leftarrow \{x : x \in \mathbf{z}, X \in \mathbf{I_G} \cup \mathbf{O_G}\}$
9: $k' \leftarrow k - |\mathbf{D}| - |\mathbf{B}| + i + 2$
10: $< \mathbf{T}, \mathbf{u_G} > \leftarrow$ HSD($\mathbf{D_G} \cup \mathbf{I_G}$, $\mathbf{u_G}$, k')
11: $\mathbf{y} \leftarrow \mathbf{y} \cup \mathbf{u_G}$, $\mathbf{D} \leftarrow \mathbf{D} \cup \mathbf{T}$
12: EVALUATE (Δ, $\mathbf{y}$), DIFFERENTIATE (Δ)
13: **else**
14: $\mathbf{D} \leftarrow \mathbf{D} \cup \{G\}$
15: $\mathbf{z} \leftarrow \mathbf{y} \cup$ IMPLICATIONS (Δ, $\mathbf{y}$)
16: $\mathbf{u_C} \leftarrow \mathbf{u_C} \cup \{x : x \in \mathbf{z}, X \in \mathbf{I_C} \cup \mathbf{O_C}\}$
17: **if** MEETSCRITERIA ($\mathbf{C}$, $\mathbf{D}$, $\mathbf{y}$) **then**
18: **return** $< \mathbf{D}$, $\mathbf{u_C} >$
19: **else**
20: goto line 3

follows when diagnosing a cone. After selecting a component with the highest probability of failure, we consider the variables of that component *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 components in the cone has been determined without knowing all the inputs to the cone (it is possible to identify a faulty component, and with strong fault models also a healthy component, without knowing all its inputs). The cost increase due to this is often insignificant because when a cone is concluded as faulty in the abstract diagnosis, the values of a significant number, if not all, of its inputs are often known.

The Algorithm Pseudocode for the hierarchical approach is given in Algorithm 2 as a recursive function. The inputs are a system $\mathbf{C}$, a set of known values $\mathbf{u_C}$ of variables at the inputs $\mathbf{I_C}$ and outputs $\mathbf{O_C}$ of the system, and again the optional integer k specifying the fault cardinality bound for the purpose of experimenting with the effect of model pruning. We start with the d-DNNF compilation of the abstraction of the given system (line 1) and then use the function SD from Algorithm 1 to get a diagnosis $\mathbf{B}$ of the abstraction (line 3), assuming that the measurement point selection and stopping criteria in Algorithm 1 have been modified according to what is described in the previous sub-section. The abstract diagnosis $\mathbf{B}$ is then used to get a concrete diagnosis $\mathbf{D}$ in a loop (lines 4–14). Specifically, if a component $G \in \mathbf{B}$ is not the root of a cone, then it is added to $\mathbf{D}$ (line 14); otherwise cone G is recursively diagnosed (line 10) and the result of it added to $\mathbf{D}$ (line 11).

Before recursively diagnosing a cone G, we compute an abnormal observation $\mathbf{u_G}$ at the inputs and output ($\mathbf{I_G} \cup \{G\}$) of G. The values of some of G's inputs and output will have been either measured or deduced. The value of a variable X is implied to be x under the measurements $\mathbf{y}$ if $Pr(X = \neg x, \mathbf{y}) = 0$, which is easy to check once Δ has been differentiated under $\mathbf{y}$. The function IMPLICATIONS(Δ, $\mathbf{y}$) (lines 7 and 15) implements this operation, which is used to compute the partial abnormal observation $\mathbf{u_G}$ (line 8). A fault cardinality bound k' for the cone G is then inferred (line 9), and the algorithm called recursively to diagnose G, given $\mathbf{u_G}$ and k'.

The recursive call returns the faults $\mathbf{T}$ inside the cone G together with the updated observation $\mathbf{u_G}$. The observation $\mathbf{u_G}$ may contain some new measurement results regarding the variables $\mathbf{I_G} \cup \{G\}$, which are added to the set of measurements $\mathbf{y}$ of the abstraction (line 11); other measurement results obtained inside the cone are ignored because internal measurements of the cone are not required in the abstraction, as explained earlier. The concrete diagnosis $\mathbf{D}$ is augmented with the faults $\mathbf{T}$ found inside the cone (line 11), and Δ is again evaluated and differentiated in light of the new measurements (line 12).

After the loop ends, the variable $\mathbf{u_C}$ is updated with the known values of the inputs $\mathbf{I_C}$ and outputs $\mathbf{O_C}$ of the system $\mathbf{C}$ (line 16). The stopping criteria are checked for the diagnosis $\mathbf{D}$ (line 17) and if met the function returns the pair $< \mathbf{D}, \mathbf{u_C} >$ (line 18); otherwise more measurements are taken until the stopping criteria (line 17) have been met.

Since $\mathbf{D}$ can contain faults from inside the cones, the compilation Δ cannot be used to check the stopping criteria for $\mathbf{D}$ (note the change in the parameters to the function MEETSCRITERIA at line 17) as the probabilistic information regarding variables inside cones is not available in Δ. The criteria are checked as follows instead: We first propagate the values of inputs in the system, and then propagate the fault effects of components in $\mathbf{D}$, one by one, by flipping their values to the abnormal ones and propagating them towards the system outputs in such a way that deeper faults are propagated first (Siddiqi and Huang 2007), and then check the values of system outputs obtained for equality with those in the observation ($\mathbf{y}$).

Example Suppose that we diagnose the abstraction of the circuit in Figure 1, with the observation $\{P = 1, Q = 1, R = 0, V = 1\}$, and take the sequence of measurements $\{D = 1, K = 1, A = 1\}$. It is concluded, from the abstract system model, that given the values of P and D, the value 1 at A is abnormal. So the algorithm concludes a fault at A. Note that $Q = 1$ and $D = 1$ suggests the presence of another fault besides A, triggering the measurement of gate B, which is also found faulty. The abstract diagnosis $\{A, B\}$ meets the stopping criteria with respect to the abstract circuit.

We then diagnose cone A, recursively, with observation $\{P = 1, D = 1, A = 1\}$. The only unknown wire J is measured and found faulty, which explains the observation at the cone's output A, given its inputs P and D. The recursion terminates and the abstract diagnosis $\{A, B\}$ generates

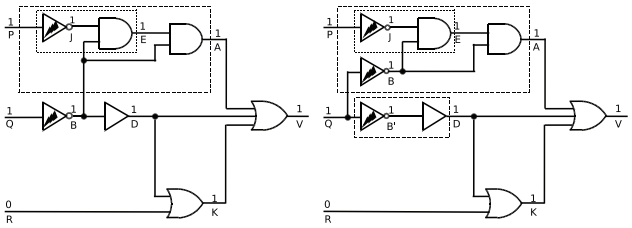

Figure 2: A faulty circuit with faults at B and J (left). Creating a clone B' of B according to D (right).

the concrete diagnosis $\{J, B\}$, which meets the stopping criteria and the diagnosis stops.

Component Cloning

In the preceding section, we have proposed an abstraction based approach to sequential diagnosis, which reduces the complexity of compilation and diagnosis by reducing the number of system components to be diagnosed. We now take one step further, aiming to handle systems that are so large that they remain intractable even after abstraction, as is the case for the largest circuits in the ISCAS-85 benchmark suite.

Our solution is a novel method that systematically modifies the structure of a system to reduce the size of its abstraction.[2] Specifically, we select a component G with parents $\mathbf{P}$ (a component X is a *parent* of a component Y, and Y is a *child* of X, if the output of Y is an input of X) that is not part of a cone and hence cannot be abstracted away in hierarchical diagnosis, and create a clone G' of it according to some of its parents $\mathbf{P'} \subset \mathbf{P}$ in the sense that G' inherits all the children of G and feeds into $\mathbf{P'}$ while G no longer feeds into $\mathbf{P'}$ (see Figure 2 for an example). We create a sufficient number of clones of G so that G and its clones become part of some cones and hence can be abstracted away. Repeated applications of this operation can allow an otherwise unmanageable system to have a small enough abstraction for compilation and diagnosis to succeed. The hierarchical algorithm is then extended to diagnose the new system and the result mapped to the original system.

We now formally define component cloning:

Definition 1 (Component Cloning). *Let G be a component in a system $\mathbf{C}$ with parents $\mathbf{P}$. We say that G is* **cloned according to parents** $\mathbf{P'} \subset \mathbf{P}$ *when it results in a system $\mathbf{C}'$ that is obtained from $\mathbf{C}$ as follows: The edges going from G to its parents $\mathbf{P'}$ are removed. A new component G' functionally equivalent to G is added to the system such that G' shares the inputs of G and feeds into each of $\mathbf{P'}$.*

In Figure 2 creating a clone B' of B according to $\{D\}$ results in a new circuit whose abstraction contains only the gates $\{A, D, K, V\}$, whereas the abstraction of the original circuit contains also gate B.

[2] Choi, Chavira, and Darwiche (2007) described a related but different technique, called *node splitting*.

Choices in Component Cloning There are two choices to be made in component cloning: Which components do we clone, and for each of them how many clones do we create and how do they split the parents?

Since the purpose is to reduce the abstraction size, clearly we only wish to clone those components that lie in the abstraction (i.e., not within cones). Among these, cloning of the root of a cone cannot reduce the abstraction size as it will destroy the existing cone, reintroducing some of the components inside the cone into the abstraction. For example, cloning D according to K in Figure 2 (right) will produce a circuit where D and its clone can be abstracted away but B' is no longer dominated (Siddiqi and Huang 2007) by D and hence is reintroduced into the abstraction. The final candidates for cloning are then precisely those components in the abstract system that are not roots of cones. The order in which components are cloned is unimportant as each when cloned will cause a reduction of precisely 1 in the abstraction size, if any.

It then remains to determine for each candidate how many clones to create and how to connect them to the parents. To understand our final method, it helps to consider a naive method that simply creates $|\mathbf{P}|-1$ clones (where $\mathbf{P}$ is the set of parents) and has each clone, as well as the original, feed into exactly one parent. Thus every parent of the component becomes the root of a cone and the component itself and all its clones are abstracted away. In Figure 2 (left), B has three parents $\{E, A, D\}$, and this naive method would create two clones of B for a total of three instances of the gate to split the three parents, which would result in the same abstraction as in Figure 2 (right).

The trick now is that the number of clones can be reduced by knowing that some parents of the component may lie in the same cone and a single clone of the component according to those parents will be sufficient for that clone to be abstracted away. In the example of Figure 2, again, the parents E, A of B lie in the same cone A and it would suffice to create a single clone of B according to $\{E, A\}$, resulting in the same, more efficient cloning as in Figure 2 (right).

More formally, we partition the parents of a component G into subsets $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_q}$ such that those parents of G that lie in the same cone are placed in the same subset and the rest in separate ones. We then create $q-1$ clones of G according to any $q-1$ of these subsets, resulting in G and all its clones being abstracted away. This process is repeated for each candidate component until the abstraction size is small enough or no further reduction is possible.

Diagnosis with Component Cloning The new system is functionally equivalent to the original and has a smaller abstraction, but is not equivalent to the original for diagnostic purposes. As the new model allows a component and its clones to fail independently of each other, it is a relaxation of the original model in that the diagnoses of the new system form a superset of those of the original. Specifically, each diagnosis of the new system that assigns the same health state to a component and its clones for all components corresponds to a diagnosis of the original system; other diagnoses are spurious and are to be ignored.

Our core diagnosis process continues to be applicable on the new system, with only two minor modifications necessary. First, the spurious diagnoses are (implicitly) filtered out by assuming the same health state for all clones (including the original) of a component as soon as the health state of any one of them is known. Second, whenever measurement of a clone of a component is proposed, the actual measurement is taken on the original component in the original system, for obvious reasons (in other words, the new system is used for reasoning and the original for measurements).

The presence of spurious diagnoses in the model can potentially skew the measurement point selection heuristic (at least in the early stages of diagnosis, before the spurious diagnoses are gradually filtered out). However, by using smaller benchmarks that could be diagnosed both with and without cloning, we conducted an empirical analysis which indicates, interestingly, that the overall diagnostic cost is only slightly affected (see next section).

Experimental Results

We now empirically evaluate our new diagnostic system, referred to as SDC (sequential diagnosis by compilation), that implements the baseline, hierarchical, and cloning based approaches. All experiments were conducted 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 2 hours and a memory limit of 1.5 GB were imposed on each test case. The d-DNNF compilation was done using the publicly available d-DNNF compiler C2D (Darwiche 2004; 2005). The CNF was simplified before compilation using the given observation, which allowed us to compile more circuits, at the expense of requiring a fresh compilation per observation.

We generated test cases for single- and multiple-fault scenarios using ISCAS-85 benchmark circuits. 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 5 test cases for each of these n scenarios. Doing the same for multiple-fault scenarios would not be practical due to the large number of combinations, so for each circuit we simply generated 500 (for circuits up to $c1355$) or 100 (for remaining circuits) random scenarios with the given fault cardinality and a random test case for each scenario.

Each test case is a faulty circuit where some gates give incorrect outputs. The inputs and outputs of the circuit are observed. The values of internal wires are then computed by propagating the inputs in the normal circuit towards the outputs followed by propagating the outputs of the assumed faulty gates one by one such that deeper faults are propagated first. The obtained values of internal wires are then 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.

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	3.6	0.01	3.4	0.01	2.8	0.01
14	GDE	3.5	6.66	3.3	15.1	3.0	14
	SDC	4.2	0.01	2.9	0.01	2.9	0.01
15	GDE	3.4	111	3.5	88	4.3	299
	SDC	3.9	0.01	3.4	0.01	3.7	0.01
16	GDE	3.3	398	3.5	556	3.2	509
	SDC	3.5	0.01	3.3	0.01	2.8	0.01
17	GDE	3.7	2876	4.6	4103	4.5	2067
	SDC	3.8	0.01	4.2	0.01	4.2	0.01

Table 1: Comparison with GDE.

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 (including the compilation time, in CPU seconds) to locate the faults, averaged over all test cases solved.

The following three subsections empirically show the effectiveness of the new heuristic, hierarchical sequential diagnosis, and component cloning, respectively.

Effectiveness of Measurement Point Selection

We first compare the baseline algorithm of SDC with a version of GDE and show that SDC performs as well in terms of diagnostic cost and scales to much larger circuits, illustrating the effectiveness of our new heuristic.

Comparison with GDE We could obtain only the tutorial version of GDE (Forbus and de Kleer 1993), which computes the set of *minimal diagnoses* instead of probable diagnoses. This makes our comparison less informative. Nevertheless, we are able to make a reasonable comparison in terms of diagnostic cost as the set of minimal diagnoses can also serve as a large set of probable diagnoses when components have equal prior probabilities. According to de Kleer et al. (1992) availability of more diagnoses aids in heuristic accuracy, whereas focusing on a smaller set of probable diagnoses can be computationally more efficient but increase the average diagnostic cost.

This version of GDE, developed for tutorial purposes, was in fact unable to solve any circuit in ISCAS-85. 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. The comparison between GDE and SDC (baseline) on these benchmarks given in Table 1 shows that SDC performs as well as GDE in terms of diagnostic cost.

Larger Benchmarks To evaluate the performance of SDC on the larger ISCAS-85 circuits, we have again conducted three sets of experiments, now involving single, double, and five faults. 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

cir-cuit	sys-tem	pru-ning	single-fault		double-fault		five-fault	
			cost	*time*	*cost*	*time*	*cost*	*time*
c432 (160 gates)	RAND	*no*	92.3	20.7	97.7	23.2	117.8	26.5
		yes	4.5	11.4	36.8	12.4	99.7	17.2
	SDC(ew)	*no*	42.0	16.6	42.5	21.3	68.4	25.5
		yes	3.7	11.1	8.6	12.0	33.8	12.8
	SDC(fp)	*no*	6.7	11.7	6.4	12.5	9.4	13.0
		yes	4.3	11.0	5.0	12.3	9.1	12.6
c499 (202 gates)	RAND	*no*	109.6	0.8	120.6	1.2	150.0	1.4
		yes	5.5	0.2	20.1	0.2	104.9	0.7
	SDC(ew)	*no*	58.1	0.7	54.0	0.5	95.8	0.8
		yes	3.6	0.2	3.7	0.2	35.7	0.3
	SDC(fp)	*no*	6.5	0.2	4.3	0.2	7.2	0.2
		yes	4.8	0.2	3.0	0.2	7.1	0.2
c880 (383 gates)	RAND	*no*	221.0	1.9	251.3	1.9	306.4	2.3
		yes	5.4	0.2	47.3	0.3	205.7	1.3
	SDC(ew)	*no*	26.8	0.3	32.8	0.4	79.0	0.7
		yes	4.0	0.2	6.8	0.2	30.5	0.4
	SDC(fp)	*no*	10.8	0.2	9.2	0.2	15.8	0.3
		yes	5.6	0.2	6.7	0.2	14.0	0.3
c1355 (546 gates)	RAND	*no*	327.2	4.3	365.7	5.7	437.4	5.6
		yes	7.4	0.4	59.0	1.0	328.6	3.5
	SDC(ew)	*no*	82.6	1.3	91.2	1.5	203.9	3.4
		yes	4.9	0.4	5.5	0.4	65.9	1.1
	SDC(fp)	*no*	34.1	0.8	14.8	0.5	19.3	0.8
		yes	8.0	0.4	9.4	0.6	18.4	0.6

Table 2: Effectiveness of measurement point selection.

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 with two different heuristics, one based on entropies of wires alone (ew) and the other based also on failure probabilities (fp). For each of the three systems we ran the same set of experiments with and without pruning the d-DNNF (using the known fault cardinality), indicated in the third column of the table. We only use the first four circuits as other circuits could not be compiled.

It is clear that the diagnostic cost is significantly lower with both heuristics of 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 and SDC-ew strategies, but has much less effect on SDC-fp except in a few cases ($c1355$ single-fault). Moreover, SDC-fp generally dominates SDC-ew, both with and without pruning.

We may also observe that (i) on the five-fault cases, SDC-fp without pruning results in much lower diagnostic cost than SDC-ew with pruning; (ii) on the double-fault cases, the two are largely comparable; and (iii) on the single-faults cases, the comparison is reversed. This indicates that as the fault cardinality rises, the combination of failure probabilities and wire entropies appears to achieve an effect similar to that of pruning. That SDC-ew with pruning performs better than SDC-fp without pruning on single-fault cases can be attributed to the fact that on these cases pruning is always exact and hence likely to result in maximum benefit.

circuit	pruning	single-fault		double-fault		five-fault	
		cost	*time*	*cost*	*time*	*cost*	*time*
c432 (64 cones)	*no*	15.4	0.4	15.8	0.5	22.2	0.5
	yes	4.9	0.3	10.4	0.4	21.5	0.4
c499 (90 cones)	*no*	7.3	0.1	5.8	0.1	10.5	0.2
	yes	4.5	0.1	3.9	0.1	9.6	0.2
c880 (177 cones)	*no*	9.5	0.1	10.2	0.1	17.4	0.2
	yes	5.6	0.1	7.6	0.1	16.3	0.2
c1355 (162 cones)	*no*	9.3	0.3	8.2	0.2	14.0	0.3
	yes	5.8	0.2	6.3	0.2	14.4	0.3
c1908 (374 cones)	*no*	11.0	222	17.1	587	34.9	505
	yes	3.0	214	8.5	463	32.4	383
c2670 (580 cones)	*no*	16.3	213	19.2	172	25.4	58
	yes	6.5	196	13.3	90	24.3	45

Table 3: Effectiveness of abstraction.

circuit	total gates	abstr. size	cloning time	total clones	abstr. size after cloning
c432	160	59	0.03	27	39
c499	202	58	0.02	0	58
c880	383	77	0.1	24	57
c1355	58	58	0.05	0	58
c1908	880	160	0.74	237	70
c2670	1193	167	0.77	110	116
c3540	1669	353	5.64	489	165
c5315	2307	385	3.6	358	266
c6288	2416	1456	0.16	0	1456
c7552	3512	545	6.68	562	378

Table 4: Results of preprocessing step of cloning.

Effectiveness of Abstraction

Table 3 reports the results of repeating the same experiments with SDC-fp using the hierarchical approach.

The running time generally reduces for all cases and we can now handle two more circuits, namely $c1908$ and $c2670$, solving 139 of 300 cases for $c1908$ (25 of single, 15 of double, and 99 of five-fault cases) and 258 of 300 cases for $c2670$ (100 of single, 60 of double, and 98 of five-fault cases). In terms of diagnostic cost, in most cases the hierarchical approach is comparable to the baseline approach. On $c432$, the baseline approach consistently performs better than the hierarchical in each fault cardinality, while the reverse is true on $c1355$.

The results indicate that the main advantage of hierarchical approach is that larger circuits can be solved. For circuits that can also be solved by the baseline approach, hierarchical approach may help reduce the diagnostic cost by quickly finding faulty portions of the circuit, represented by a set of faulty cones, and then directing the measurements inside them, which can result in more useful measurements. On the other hand, it may suffer in cases where it has to needlessly go through hierarchies to locate the actual faults, while the baseline version can find them more directly and efficiently. Finally, we note that pruning helps further reduce the diagnostic cost to various degrees.

circuit	single-fault		double-fault		five-fault	
	cost	*time*	*cost*	*time*	*cost*	*time*
c432	7.2	10.3	6.6	7.8	9.6	9.7
c880	11.2	0.2	9.3	0.2	16.2	0.3

Table 5: Effect of component cloning on diagnostic performance.

circuit	single-fault		double-fault		five-fault	
	cost	*time*	*cost*	*time*	*cost*	*time*
c432	15.2	0.1	14.8	0.1	20.2	0.1
c880	8.8	0.1	9.3	0.1	15.8	0.2
c1908	13.6	2.8	18.3	5.0	35.4	5.1
c2670	13.5	4.5	15.3	0.7	20.1	2.3
c3540	27.8	382	30.5	72.5	36.1	108.6
c5315	7.2	2.5	21.1	5.9	24.4	6.6
c7552	70.6	1056	43.1	129.0	104.8	1108

Table 6: Effectiveness of component cloning ($c499$ and $c1355$ omitted as they are already easy to diagnose and cloning does not lead to reduced abstraction).

Effectiveness of Component Cloning

In this subsection we discuss the experiments with component cloning. We show that cloning does not significantly affect diagnostic cost and allows us to solve nearly all the circuits in the ISCAS-85 suite.

Table 4 shows the result of the preprocessing step of cloning on each circuit. The columns give the name of the circuit, the total number of gates in that circuit, the size of the abstraction of the circuit before cloning, the time spent on cloning, the total number of clones created in the circuit, and the abstraction size of the circuit obtained after cloning. On all circuits except $c499$, $c1355$, and $c6288$, a significant reduction in the abstraction size has been achieved. $c6288$ appears to be an extreme case with a very large abstraction that lacks hierarchy, while gates in the abstractions of $c499$ and $c1355$ are all roots of cones, affording no opportunities for further reduction (note that these two circuits are already very simple and easy to diagnose).

We start by investigating the effect of component cloning on diagnostic performance. To isolate the effect of component cloning we use the baseline version of SDC (i.e., without abstraction). Table 5 summarizes the performance of baseline SDC with cloning on the circuits $c432$ and $c880$. Comparing these results with the corresponding entries in Table 2 shows that the overall diagnostic cost is only slightly affected by cloning. We further observed that in a significant number of cases, the proposed measurement sequence did not change after cloning, while in most of the other cases it changed only insubstantially. Moreover, in a number of cases, although a substantially different sequence of measurements was proposed, the actual diagnostic cost did not change much. Finally, note that the diagnosis time in the case of $c432$ has reduced after cloning, which can be ascribed to the general reduction in the complexity of compilation due to a smaller abstraction.

Results in Table 6 illustrate the performance of hierarchical sequential diagnosis with component cloning—the most scalable version of SDC. All the test cases for circuits $c1908$ and 2670 were now solved, and the largest circuits in the benchmark suite could now be handled: All the cases for $c5315$, 164 of the 300 cases for $c3540$ (34 of single-, 65 of double-, and 65 of five-fault cases), and 157 of the 300 cases for $c7552$ (60 of single-, 26 of double-, and 71 of five-fault cases) were solved. In terms of diagnostic cost, cloning generally resulted in a slight improvement. In terms of time, the difference is insignificant for $c432$ and $c880$, and for the larger circuits ($c1908$ and $c2670$) diagnosis with cloning was clearly more than an order of magnitude faster.

Related Work

The idea of testing the most likely failing component comes from (Heckerman, Breese, and Rommelse 1995), where the testing of a component was considered a unit operation. The heuristic was computed assuming a single fault (this assumption could compromise the diagnostic cost in multiple-fault cases as the authors pointed out). In our case, by contrast, the testing of each variable of a component is a unit operation, calling for a more complex heuristic; also, we do not need to assume a single fault. Our work also goes further in scalability using several structure-based techniques: compilation, abstraction, and component cloning.

In the GDE framework, de Kleer (2006) studied the sensitivity of diagnostic cost to the *ϵ-policy*, which provides estimates for the probabilities of diagnoses. In our case, probabilities of diagnoses are not required, and those probabilities required can be computed exactly and efficiently.

Flesch, Lucas, and van der Weide (2007) proposed a new framework to integrate probabilistic reasoning into model-based diagnosis. However, they did not address the problem of sequential diagnosis.

Most recently, Feldman, Provan, and van Gemund (2009) proposed a related method for reducing diagnostic uncertainty. While our work attempts to identify the actual faults with the fewest individual measurements, their heuristic was aimed at reducing the number of diagnoses with the fewest test vectors.

Conclusion

We have presented a new system for sequential diagnosis, called SDC, that employs three new structure-based techniques—a new more efficient heuristic for measurement point selection, abstraction-based sequential diagnosis, and component cloning—to scale diagnosis to larger systems with low diagnostic costs.

Topics for ongoing and future work include extensions to cases where measurements have varying costs, the feasibility of finding optimal measurement selection policies, and the application of the proposed techniques to model-based testing and to other probabilistic queries (such as MPE and MAP) on Bayesian networks.

Acknowledgments

We thank the anonymous reviewers for their comments. An initial part of this work has been presented at a symposium (Siddiqi and Huang 2008). NICTA is funded by the

Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Choi, A.; Chavira, M.; and Darwiche, A. 2007. Node splitting: A scheme for generating upper bounds in Bayesian networks. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 57–66.

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. *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. 1992. Focusing on probable diagnosis. In *Readings in model-based diagnosis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 131–137.

de Kleer, J. 2006. Improving probability estimates to lower diagnostic costs. In *17th International Workshop on Principles of Diagnosis (DX)*.

Feldman, A.; Provan, G.; and van Gemund, A. 2009. FRACTAL: Efficient fault isolation using active testing. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 778–784.

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.

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.

Siddiqi, S., and Huang, J. 2008. Probabilistic sequential diagnosis by compilation. In *International Symposium on Artificial Intelligence and Mathematics*.